Journal of Trends in Computer Science and Smart Technology (ISSN: 2582-4104)
www.irojournals.com/tcsst/

Neuroplastic Fuzzy GCN-LSTM: A Hybrid Graph-based Deep Learning Model for Accurate Battery SoC Prediction

Munish Kanna S.[1], Narmadha G.[2]

[1]Research Scholar, [2]Associate Professor, Department of Electrical and Electronics Engineering, Sethu Institute of Technology, Pulloor, Kariapatti, India.

Email: [1]munishkanna742@gmail.com, [2]gnarmadhame@gmail.com

Abstract

Accurate prediction of the State of Charge (SoC) in batteries is critical for the efficient and safe operation of Electric Vehicles (EVs). In this work, a novel hybrid neural architecture called Neuroplastic Fuzzy GCN-LSTM is proposed for accurate SoC prediction. It involves fuzzy logic, dynamic graph modelling and graph convolutional networks (GCNs) to effectively handle the complex patterns of battery data. Initially, time-series sensor inputs of voltage, current, and temperature are normalised and transformed into fuzzy linguistic representations to address uncertainty and improve interpretability. Then, a dynamic graph is constructed to capture inter-feature dependencies via a Gaussian similarity kernel. These graphs are processed through spectral GCN layers to extract spatial correlations. Finally, the neuroplastic LSTM (NP-LSTM) is applied for SoC prediction. Unlike conventional LSTMs, the NP-LSTM adaptively modulates its memory cell updates using error-based synaptic plasticity, allowing the model to emphasise learning from past prediction errors. The performance of the Neuroplastic Fuzzy GCN-LSTM model is verified using NASA's Prognostics Center of Excellence data sets in terms of Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Coefficient of Determination (R^2) compared to models. The Neuroplastic Fuzzy GCN-LSTM model achieved the best performance among all competing models, with the lowest MSE (0.009378), MAE (0.086234), and RMSE (0.096839).

Keywords: EV, SoC, Neuroplastic Fuzzy GCN-LSTM, Prediction, MSE.

1. Introduction

Electric vehicles (EVs) are becoming increasingly popular due to the worldwide trend toward green transport [1]. It is projected that the worldwide EV market will expand to more than 200 million automobiles by 2030, driven by the enforcement of stringent emission controls by governments and the decline in the price of lithium-ion batteries with advancing technology [2]. The battery system lies at the heart of all EVs and has a significant impact on user satisfaction, the cost-effectiveness of EV rollout, and vehicle range and safety.

Estimation of SoC is one of the most critical parts of EV battery management systems. SoC in a battery is a measure of its residual energy [3]. Precise SoC estimation is crucial for applications such as energy optimization, charge scheduling, and preventing over-discharge and over-charge of batteries. Traditional SoC estimation techniques include open circuit

Received: 11.07.2025, received in revised form: 10.08.2025, accepted: 23.08.2025, published: 02.09.2025

voltage (OCV)-based models and coulomb counting. However, these methods do not consider sensor drift or non-generalizability across loads and environmental conditions.

Machine learning (ML) and deep learning (DL) algorithms have become increasingly popular in the last few years because they can identify non-linear interactions and learn from experience [4][5]. Machine learning (ML) algorithms, such as support vector machines (SVMs), decision trees, and AdaBoost, have been found to identify complex features in battery data. DL models, specifically RNNs and their extensions, such as LSTM and GRUs, are usually employed to learn temporal dependencies in time-series data. The models perform better than the conventional methods in real-time EV settings [6]. Since they process the input features independently, conventional LSTM models may fail to identify intricate inter-feature interactions or adapt to unanticipated and dynamic operating conditions. In addition, there is a possibility that certain static architectures employed in existing models may not be optimal for non-stationary operating conditions. Finally, there is less work on handling sensor uncertainty and implementing fuzzy logic in graph-based feature interaction modeling.

To bridge this gap, a novel modified hybrid LSTM architecture called Neuroplastic Fuzzy GCN-LSTM is proposed in this work. The key innovations of the proposed model are:

- Fuzzy Membership Encoding: Enhances feature robustness and interpretability by converting continuous inputs into fuzzy linguistic categories.
- Dynamic Graph Construction: Models the time-varying dependencies between features using a Gaussian similarity kernel. It generates an adaptive feature graph for each time step.
- Graph Convolutional Network (GCN): Captures spatial (inter-feature) relationships via spectral graph convolution to enrich temporal features.
- Neuroplastic LSTM: Introduces plasticity-inspired error correction mechanisms that mimic synaptic updates in biological systems. It allows memory units to adapt based on past prediction errors.

The rest of the paper is structured as follows: Section 2 presents the existing work related to SoC prediction, and Section 3 presents the proposed prediction model. Section 4 explains the datasets, the experimental results, and the evaluation metrics. Section 5 concludes the paper.

2. Related Work

A dual adaptive Kalman filtering algorithm-based SoC prediction technique is proposed by P. Reshma et al [7]. The design parameters of batteries are optimized using the Remora Optimization Algorithm. Likewise, Li, D.C. et al. [8] proposed a DL model combined with an extended Kalman filter (EKF) approach for SoC prediction. The accuracy rate of the model is higher than that of sole DL models. Similarly, Ravish Yadav et al. [9] developed an SoC estimation model using the Augmented Adaptive Extended Kalman Filter (AAEKF) technique. Additionally, a control technique is introduced to increase prediction accuracy. Results on the LiFePO4 battery dataset show that the AAEKF technique results in less than a 3% error rate compared to existing Kalman filter techniques.

To learn spatiotemporal patterns in battery SoC data, a hybrid fusion model is proposed by Sadiqa Jafari et al [10]. The hybrid model combines LSTMs with Convolutional LSTM (ConvLSTM) architectures to accurately learn the historical data of the battery. In addition, the parameters of the model are altered using Particle Swarm Optimization (PSO). Results show that the proposed model achieves a MAE of 0.03, a RMSE of 0.085, and an R^2 score of 93%, respectively.

Basak Gok et al. [11] analyzed the performance of ML models like Neural Networks (NN) and Decision Trees for the estimation of SoC. The SoC rates are predicted under dynamic and sub-zero conditions. Results on a real-time dataset show that both models achieve high R^2 values between 0.98 and 1.

To consider both data-temporal dependency and unpredictability in the battery's time series, a dual LSTM model is proposed by Junyoung Ahn et al [12]. This LSTM model consists of a mainstream (m–) LSTM and a gradient (g–) LSTM unit to extract hidden features. Compared to the vanilla LSTM, the proposed model shows a 9.2% accuracy improvement in SoC prediction.

Jayaraman, R. et al. [13] propose a two-fold model for SoC prediction. Initially, the features from battery series data are extracted using the Moore–Penrose Pseudo-Inverse Principal Component Analysis (MPPI-PCA). Then, the extracted features are fed into the LSTM model for prediction. The integration of MPPI-PCA reduces the error rate considerably when compared to sole LSTM models.

In their study, Uzair Khan et al. [14] presented an SoC estimation approach using self-attention DL models. The self-attention model combines both non-recurrent and recurrent architectures to process the time series data. Simulation results on the 18650PF dataset show that the DNN achieves MAE $< 1\%$ and RMSE $\approx 1\%$ across datasets at temperatures of 9°C, 35°C, and 55°C, respectively. Similarly, Saad El Fallah et al. [15] proposed a DNN model and compared its performance with Gated Recurrent Unit (GRU) and RNN models. Results were verified on Li-ion battery 18650 datasets. The DNN model achieves a mean error of less than 0.5% and outperforms GRU and RNN in capturing SoC under varying temperatures and dynamic profiles.

B. Devi et al. [16] propose a Cloud-Integrated Battery Management System (CIBMS) using IoT technology. The real-time battery data of voltage, current, and temperature is collected via Raspberry Pi 4B+ and processed using a DL model for SoC prediction in EVs. Among different DL models, the CNN-GRU-LSTM model achieves the lowest MSE and RMSE rates.

Muhammad Hamza Zafar et al. [17] propose a Hybrid Multi-Layer DL model for real-time SoC prediction in Evs. Additionally, the parameters are tuned using the Mountain Gazelle Optimizer (MGO). Experimental results on four Li-ion battery datasets (DST, BJDST, FUDS, US06) show that the HMDNN model achieves an average NMSE of 0.1% and RMSE of 0.3% for the test sets.

Tian, H. et al. [18] proposed an ensemble Support Vector Regression (SVR) model for SoC prediction. Initially, the dataset is separated into multiple subsets. Then, the SVR model is applied to each subset. The final SoC is obtained by combining all the results.

To solve the issues in RNN models, Junxiong Chen et al. [19] proposed a modified LSTM model for SoC prediction. An additional input, computed based on the sliding window average voltage, is added to the conventional LSTM model to increase prediction accuracy. Results on $LiFePO_4$ battery datasets show that the modified LSTM achieves RMSE < 1.3% and MAXE < 3.2% for varying temperatures.

The ML model of Extra Tree Regressor (ETR)-based SoC prediction is proposed by Sadiqa Jafari et al. [20] for EV batteries. In addition, a PSO is applied to adjust the ETR parameters. Compared to LightGBM, the ETR model shows an R^2 of 0.9983, RMSE of 0.62, MAE of 0.085, and MSE of 0.39 for the 30% test set.

A hybrid model of the GRU-RNN-based SoC prediction model is proposed by Edward Ositadinma Ofoegbu et al [21]. The performance of the model is analysed for different layer counts and configurations. Shih-Lin Lin et al. propose a deep RNN model for EV battery SoC estimation. By considering environmental, vehicle, and battery parameters, the proposed model's performance is analysed for different datasets of real-world driving data from a BMW i3.

Xiaogang Wu, et al [22] proposed a two-stage SoC prediction model for the travel data of EVs in Beijing. To reduce the dimensionality of the data, the ML model of Random Forest (RF) is used. Then, the LSTM model is applied for final predictions. Compared to GRU and RNN models, the proposed RF combined LSTM model reduces MSE by 8.5%. Similarly, Mohd Herwan Sulaiman et al [23] proposed an RF-based SoC prediction approach for real-world EV SoC predictions. The RF model is applied in the BMW i3 battery dataset and compared with the Extreme Learning Machine (ELM) model. In comparison, RF shows a lower RMSE of 4.7% compared to 5.4% for ELM, and a lower MAE of 3.3441% versus 4.5% for ELM for validation sets.

To predict EV battery SoC more accurately, a Graph Convolutional Network (GCN)-based prediction model is proposed by Kim, G et al [24]. The model uses learnable graphs to capture relationships between variables and time. It improves both prediction accuracy and interpretability using real-world EV data. In summary, existing SoC prediction models are limited by their inability to handle sensor uncertainty and failure to model inter-feature dependencies. In addition, this model is fully based on static architectures and lacks integration between temporal and spatial dependencies.

3. Proposed Methodology: Neuroplastic Fuzzy GCN-LSTM for SoC Prediction

The proposed system is a novel hybrid neural architecture that integrates fuzzy logic, dynamic graph modeling, graph convolutional encoding, and neuroplastic LSTM memory mechanisms to improve the prediction of SoC in batteries. The layers of the Neuroplastic Fuzzy GCN-LSTM model are shown in Figure 1.

Input Representation and Preprocessing

The input to the model consists of time series sensor readings extracted from battery discharge/charge cycles. Each time step t includes features such as voltage Vt, current It, and temperature Tt. The raw inputs xt=[Vt, It, Tt] are first normalised using Min-Max scaling to the range [0,1] to ensure numerical stability (Eq. 1):

$$x_t^{norm} = \frac{x_t - \min(x)}{\max(x) - \min(x)} \tag{1}$$

These values are then segmented into fixed-length sequences of length L using a sliding window approach, forming a tensor X∈R $^{N×L×F}$ where N is the number of sequences and F is the number of features.

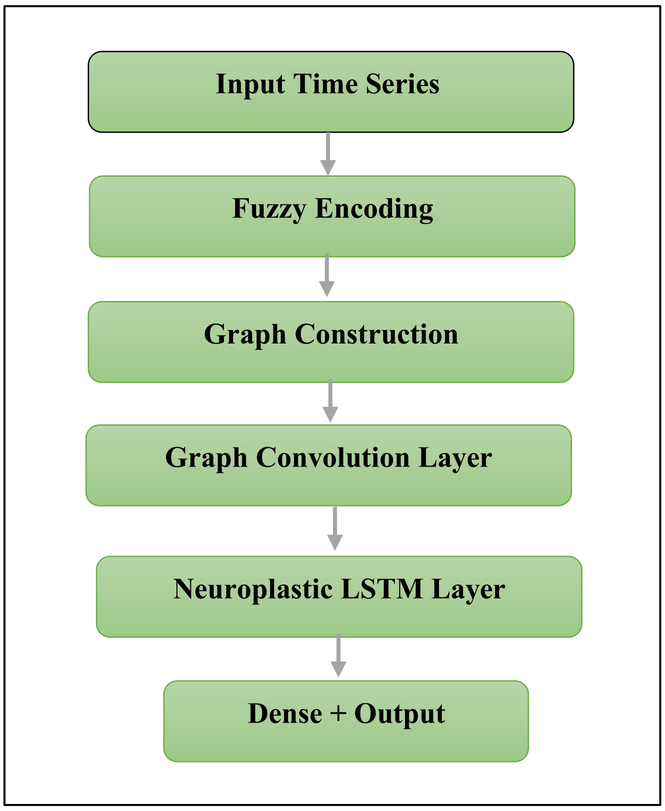


Figure 1. Neuroplastic Fuzzy GCN-LSTM Layers

Fuzzy Membership Encoding

To introduce interpretability and tolerance to uncertainty, each feature is passed through a Fuzzy Membership Layer. In the proposed Neuroplastic Fuzzy GCN-LSTM model, the fuzzy logic system uses Gaussian membership functions. Each input value is projected into linguistic categories: Low, Medium, and High using Gaussian membership functions (Eq. 2):

$$\mu_k(x) = \exp\left(-\frac{(x-c_k)^2}{2\sigma_k^2}\right) \tag{2}$$

Where ck and σk represent the center and spread of the k-th fuzzy set. This transforms the original input vector xt∈ R^F into a higher-dimensional fuzzy vector x~t∈R $^{F×K}$, where K is the number of fuzzy classes. This step increases the model's ability to generalize across data uncertainties and introduces linguistic interpretability to the learned features. The membership functions are derived based on domain knowledge and are optimized using the model's training process. In the proposed system, the parameters of the fuzzy membership functions are learned during training.

Dynamic Graph Construction

The graph topology for the Graph Convolutional Network (GCN) is defined by the adjacency matrix which captures the relationships between nodes (features) in the graph. A dynamic graph is constructed at each time step using fuzzy features to model inter-feature dependencies. Nodes in the graph represent input features (e.g., fuzzy voltage, current), and edges represent their pairwise relationships which are captured using a Gaussian similarity kernel as follows (Eq. 3):

$$A_{i,j} = \exp\left(-\frac{||\tilde{x}_t^i - \tilde{x}_t^j||^2}{\sigma^2}\right) \tag{3}$$

This results in an adjacency matrix A∈R $^{F\times F}$, capturing temporal and spatial correlations among fuzzy-encoded features. Unlike static graphs, this adjacency matrix is dynamic. It is computed afresh for each sequence window. It is mainly used for the model to adapt to changing signal relationships over time.

Graph Convolutional Encoding

To extract topological features from the constructed graph, we apply GCN over the fuzzy-encoded graph. We use the spectral GCN formulation (Eq. 4):

$$H^{(l+1)} = \sigma(\widetilde{D}^{-1/2}\tilde{A}\widetilde{D}^{-1/2}H^{(l)}W^{(l)} \tag{4}$$

Where, H(l)) is the input feature matrix at layer l, W(l) is the weight matrix, $\tilde{A}$=A+I is the adjacency matrix with added self-loops, $\widetilde{D}$ is the degree matrix of $\tilde{A}$, σ is a non-linear activation function. The GCN layers transform the input feature graph into spatially rich embeddings for better context regarding feature interactions.

The fuzzy logic parameters and graph structures are both learned dynamically during training to adapt to the data. The input features and signal correlations are predefined based on the physical features of the battery system.

Neuroplastic LSTM Layer

The LSTM network is a category of RNN specifically developed to solve the vanishing gradient issues of standard RNNs. LSTM includes memory cells and gating mechanisms to regulate the flow of information across time steps. This process is used for the model to retain long-term dependencies in sequential data. An LSTM unit consists of the following key gates:

- Forget Gate: Determines what data from the previous cell state should be forgotten.
- Input Gate: Controls what new data will be added to the cell state.
- Candidate Memory: The new candidate values to be added.
- Output Gate: Decides the output based on the updated memory.

The operation of the LSTM cell is described by the following equations (Eq. 5 - Eq. 10):

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{5}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{6}$$

$$\widetilde{C}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{7}$$

$$C_t = f_t \odot C_{t-1} + i_i \odot \widetilde{C}_t \tag{8}$$

$$o_t = \sigma(W_o x_t + U_O h_{t-1} + b_o) \tag{9}$$

$$h_t = o_t \odot \tanh(C_t) \tag{10}$$

Where, x_t is input vector at time step t, h_{t-1} is a hidden state from the previous time step, C_t is the memory cell, σ is the sigmoid activation function, ⨀ is the element-wise multiplication

In traditional LSTM fixed update rule is used for memory update. In Neuroplastic LSTM, a dynamic correction strategy is used which works based on the model's own prediction error. It simulates biological synaptic plasticity.

Inspired by Hebbian learning and adaptive memory modulation, the Neuroplastic LSTM updates its cell state to give higher importance to time steps where previous predictions were inaccurate. This process is used to focus on learning with new input patterns. To this, we introduce a neuroplastic correction using a prediction error ϵt (Eq. 11):

$$\epsilon_t = |x_t - \hat{x}_{t-1}| \tag{11}$$

This error term is incorporated into the cell update with a plasticity modulation matrix P (Eq. 2):

$$C_t^{new} = c_t + P \odot \epsilon_t \tag{12}$$

Where P is a learned parameter that mimics Hebbian learning. This allows the model to focus its memory updates where it made large previous prediction errors.

Dense Output Layer and Loss Function

The final hidden state h_t From the NP-LSTM is passed through one or more Dense layers to generate the predicted SoC (Eq. 13):

$$\hat{y}_{t=Dense(h_t)} \tag{13}$$

The model is trained using the MSE loss between the predicted. $\hat{y}_t$and actual y_t SoC values (Eq. 14):

$$L = \frac{1}{N}\sum_{t=1}^{N}(\hat{y}_t - y_t)^2 \tag{14}$$

Additional metrics like MAE and R^2 Score are used to evaluate prediction accuracy.

Compared to other simpler models, the combination of neuroplasticity, fuzzy logic, GCN, and LSTM is chosen due to their complementary strengths. Neuroplasticity is used for dynamic memory updates based on past errors, which improves learning. In addition, fuzzy logic handles uncertainty, and GCN models feature interactions across time-varying data. Also, LSTM captures long-term dependencies to increase prediction accuracy.

Coati Optimization Algorithm (COA)

The Coati Optimization Algorithm (COA) is motivated by the natural foraging and survival behaviors of coatis, omnivorous mammals native to the Americas [26][27]. These animals exhibit intelligent group strategies during hunting (particularly iguanas) and during predator evasion. The dual-phase behavioral modelling of coatis—(1) hunting iguanas and (2) escaping predators forms the foundation of the COA and is mathematically formulated to guide the optimization process.

Mathematical Formulation of COA

Initialization Phase

The optimization begins by initializing a population of coatis randomly within the search area. Each coati represents a candidate solution Xi=[xi,1,xi,2,...,xi,m], where m is the number of decision variables. The initialization equation is (Eq. 15):

$$x_{i,j} = lb_j + r.\left(ub_j - lb_j\right),\ \ i = 1,2 \ldots\ldots m \tag{15}$$

Here, r∈[0,1] is a uniformly distributed random number, and lb_j, ub_j are the lower and upper bounds of the jth variable, respectively. The fitness of each coati is evaluated using an objective function F(Xi)) which forms the population matrix as follows (Eq. 16):

$$X = \begin{pmatrix} x_{i,j} & . & . & x_{1,m} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ x_{N,1} & . & . & x_{N,m} \end{pmatrix}, F = \begin{pmatrix} F(X_1) \\ . \\ . \\ F(X_N) \end{pmatrix} \tag{16}$$

Phase I: Exploration (Hunting Iguanas)

In nature, coatis demonstrate cooperative behavior when hunting iguanas. A subset climbs trees to scare iguanas and others wait below to capture them. This behavior is modelled to diversify the search (exploration):

Tree-Climbing Group:

A group of ⌊N/2⌋ coatis move toward the best-known solution (iguana's position) (Eq. 17)

$$x_{i,j}^{P1} = x_{i,j} + r.\left(I_{guna_j} - I.x_{i,j}\right) \tag{17}$$

where I_{guna_j} is the coordinate of the current global best solution, and I∈{1,2} is a random integer.

Ground-Attack Group:

The remaining coatis respond to a randomly repositioned iguana (exploration through randomness) (Eq. 18 & Eq. 19):

$$I_{guana}G_j = lb_j + r.\left(ub_j - lb_j\right) \tag{18}$$

$$x_{i,j}^{P1} = \begin{cases} x_{i,j} + r.\left(I_{guana}G_j - I.x_{i,j}\right),\ if\ F\left(I_{guana}G\right) < F_i \\ x_{i,j} + r.\left(x_{i,j} - I_{guana}G_j\right),\ otherwise \end{cases} \tag{19}$$

Each candidate's update is conditional on fitness improvement (Eq. 20):

$$X_i = \begin{cases} X_i^{P1}, & if\ F\ (X_i^{P1}) < F_i \\ X_i, & otherwise \end{cases} \tag{20}$$

This phase increases global search capacity and prevents early convergence by encouraging diversity.

Phase II: Exploitation (Escaping Predators)

When confronted by predators, coatis exhibit evasive behavior by moving slightly away from their current positions. This behavior is translated into a local search strategy to exploit promising regions:

Local Bound Scaling:

Local bounds are dynamically reduced based on the iteration count t, narrowing the search as the algorithm progresses (Eq. 21):

$$lb_j^{local} = \frac{lb_j}{t}, ub_j^{local} = \frac{ub_j}{t} \tag{21}$$

Local Position Update:

Each coati perturbs its position slightly within local bounds (Eq. 22):

$$x_{i,j}^{P2} = x_{i,j} + (1 - 2r).\left(lb_j^{local} + r(ub_j^{local} - lb_j^{local})\right), \tag{22}$$

Again, the new position is retained only if it improves the objective (Eq. 23):

$$X_i = \begin{cases} X_i^{P2}, & if\ F\ (X_i^{P2}) < F_i \\ X_i, & otherwise \end{cases} \tag{23}$$

This phase ensures fine-grained exploitation and convergence to an optimal or near-optimal solution.

Parameter Tuning Using COA

To increase the predictive capability of the Neuroplastic LSTM, the COA is used to optimally tune the critical hyperparameters of the model. The hyperparameters tuned using COA include the count of LSTM units, Batch Size (BS), Learning Rate (LR), and the neuroplasticity modulation rate (η). The objective of the optimization process is to minimize the prediction error on validation data, which is quantitatively measured using the MSE as the fitness function. The MSE is given by (Eq. 24):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{24}$$

where y_i and $\hat{y}_i$ represent the real and predicted outputs, respectively, and n is the total number of samples. During each iteration of the COA, new parameter sets are generated and evaluated using this fitness function. The best-performing parameters are retained and propagated across generations through cooperative learning and social guidance mechanisms inherent to the COA.

Pseudocode: Coati Optimization for Neuroplastic LSTM Hyperparameter Tuning

```
Input:
            - Dataset D = {X_train, y_train, X_val, y_val}
            - Search space S = {units_range, batch_sizes, learning_rates, η_range}
            - Max_iterations, Population_size
Output:
            - Best_params (optimal LSTM parameters)
            - Best_model (trained Neuroplastic LSTM model)
1. Initialize a population of coatis with random hyperparameters from S
           For each coati_i in population:
              coati_i.position = random_parameters(S)
2. Evaluate fitness (MSE) of each coati
           For each coati_i:
              Train Neuroplastic LSTM with coati_i.position
              Evaluate on validation set
              coati_i.fitness = MSE(y_val, y_pred)
3. Identify the best coati (global_best) with the lowest MSE
4. For iteration = 1 to Max_iterations do:
            For each coati_i in population:
               - Generate a new solution (position) based on:
                  a) Local search (exploration): small random changes
                  b) Global guidance (exploitation): toward global_best
                     - Clip or adjust parameters within bounds of S
               - Train Neuroplastic LSTM with new_position
               - Evaluate MSE on validation set
               - If new MSE < coati_i.fitness:
                  coati_i.position = new_position
                  coati_i.fitness = new_MSE
                     - If new_MSE < global_best.fitness:
                  global_best = coati_i
5. Return best_params = global_best.position, best_model = model trained with best_params
```

In the first step, an initial population of coatis is generated, where each coati represents a candidate solution with randomly selected hyperparameters from the search space SSS. Each coati's fitness is then evaluated by training a Neuroplastic LSTM model using its specific parameter set and calculating the Mean Squared Error (MSE) on the validation data. This MSE serves as the fitness score, where a lower value indicates better predictive performance.

The algorithm identifies the best-performing coati—the one with the lowest MSE and designates it as the global_best. Subsequently, the optimization process enters an iterative phase, running for a specified number of iterations. In each iteration, every coati generates a new solution by either performing a local search or following a global guidance strategy.

Each new solution is evaluated by training a new Neuroplastic LSTM model and computing its MSE. If the new MSE is better than the current coati's fitness, the coati updates its parameter set accordingly. Additionally, if the new solution outperforms the global best, the global_best is updated with this new coati's parameters. After completing the specified number of iterations, the algorithm returns the best set of LSTM hyperparameters found during the search and the corresponding trained Neuroplastic LSTM model.

4. Result and Discussion

The proposed model is implemented using the Python programming language and executed in the IDLE environment. The dataset used for simulation is obtained from NASA's Prognostics Center of Excellence. It provides real-time data collected from lithium-ion batteries undergoing repeated charge and discharge cycles under varying operational conditions. Specifically, the datasets used in this study include B0005, B0006, B0007, B0018, and B0055, etc. Figures 2 and 3 visualise the sample charging and discharging patterns of the B0005 battery. Each dataset is stored in .mat format and contains a MATLAB structure comprising multiple battery cycles recorded at different time intervals throughout the battery's operational life. Each data entry includes the following fields:

- cycle: The sequential number of the charge/discharge cycle (integer);
- voltage_measured: The terminal voltage of the battery in volts (V);
- current_measured: The current drawn from the battery in amperes (A);
- temperature_measured: The surface temperature of the battery in degrees Celsius (°C);
- voltage_load: The voltage measured under load conditions (V);
- current_load: The current measured under load conditions (A);
- time: The timestamp of each measurement in seconds;
- capacity: The remaining battery capacity in ampere-hours (Ah), which serves as the target variable for prediction.

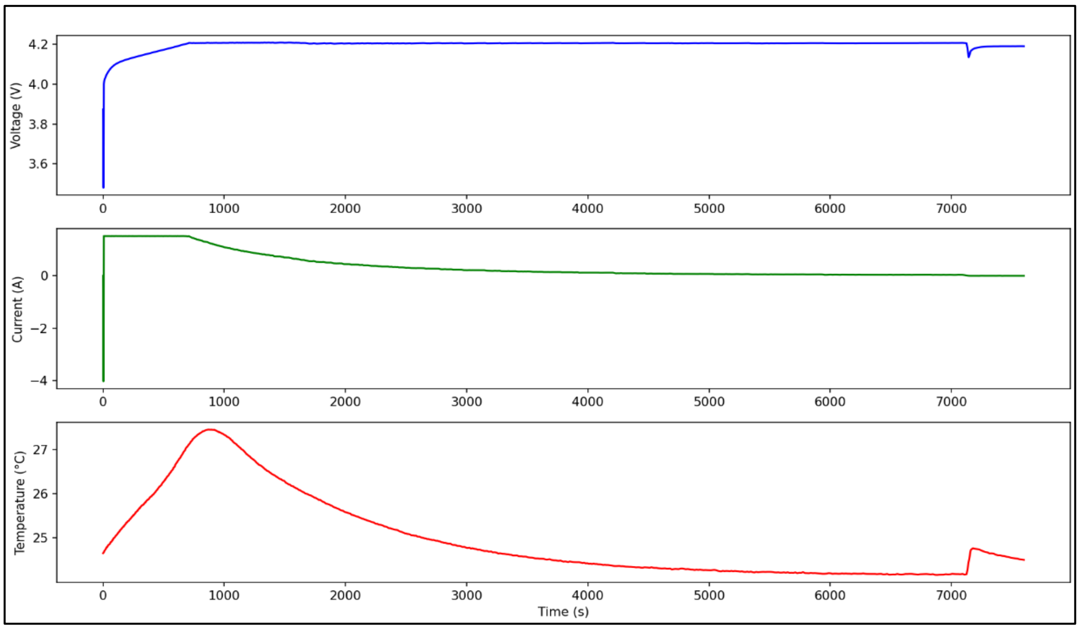


Figure 2. Charging Performance of B0005

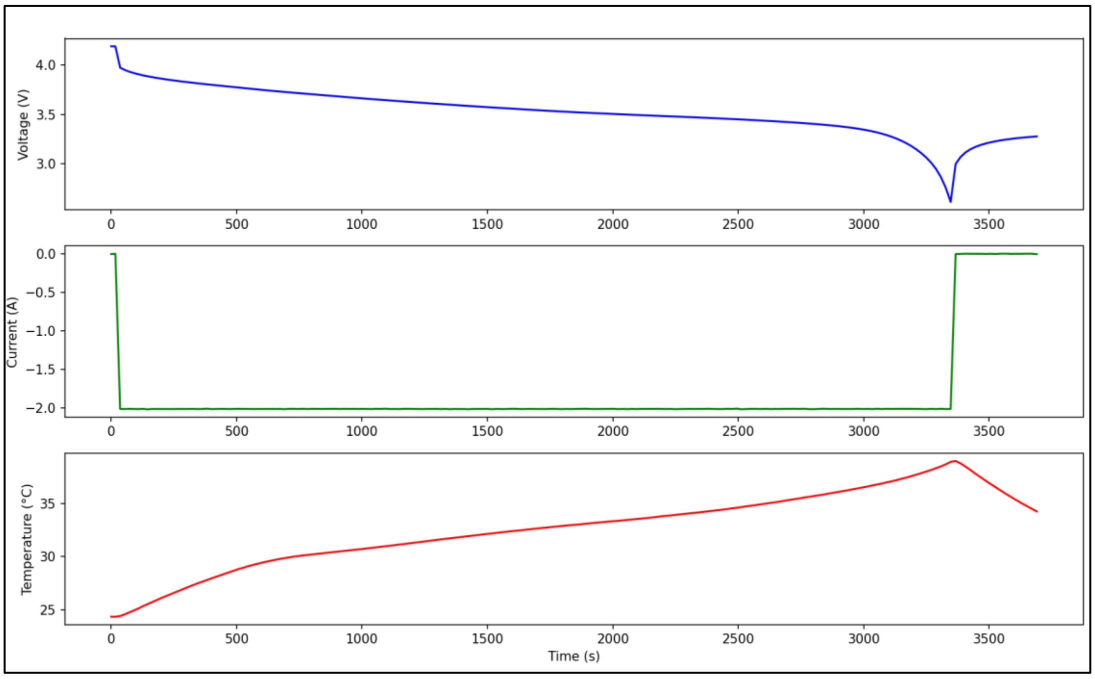


Figure 3. Discharging Performance of B0005

To ensure the dataset is suitable for input into machine learning models, all feature values were normalized to a range of [0, 1] using the MinMaxScaler technique. This normalization process is used to improve convergence during training and ensure uniform feature scaling. To effectively capture temporal dependencies in the battery degradation behavior, a sliding window approach is used. This method generates sequential input samples by grouping time-series data over fixed-size windows. For a fair evaluation of the model's predictive performance, the dataset was split into 80% for training and 20% for testing.

Initially, default hyperparameters are chosen based on commonly used configurations in time series forecasting models. However, these are subsequently fine-tuned using COA which efficiently explored the parameter space to minimize MSE on the validation dataset. Table 1 below summarises the initial and optimized values of key parameters.

Table 1. Optimized Key Parameters

Parameter	**Initial Value**	**Optimized Value**
LSTM Units	64	123
BS	16	32
LR	0.001	0.000221
Neuroplasticity (η)	1.0	0.5

The fitness curve of COA is given in Figure 4. It shows how the COA algorithm progressively improves the model's performance from an initial state until it reaches a point of optimal or near-optimal hyperparameter configuration. The low MSE value represents the best performance achieved by the fine-tuned hyperparameters.

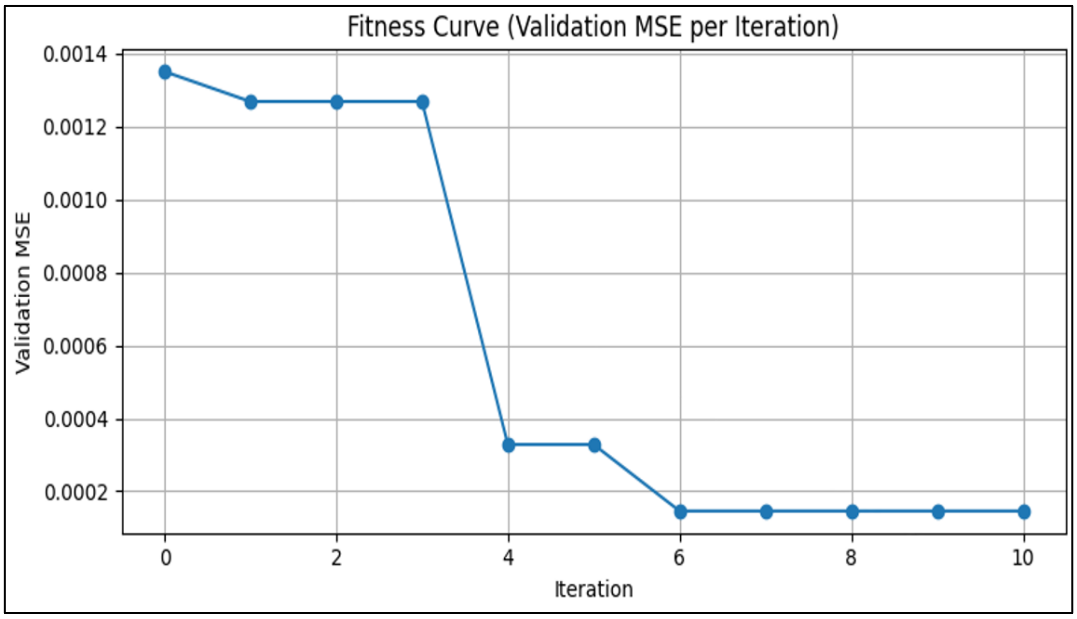


Figure 4. Fitness Curve of COA

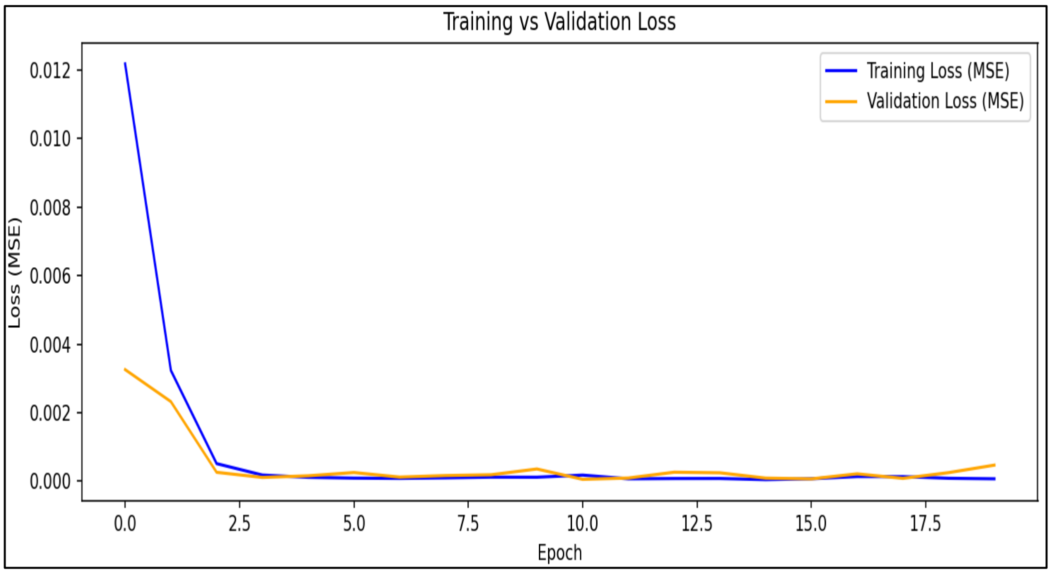


Figure 5. Model Validation Curve

The validation curve of the proposed model is given in Figure 5. The rapid reduction in both training and validation loss in the epochs specifies that the model is learning successfully from the data and improving its performance. The performance of the proposed Neuroplastic Fuzzy GCN-LSTM model is compared against several baseline models like LSTM, GRU, RNN, Dense Neural Network, and Random Forest. The metrics used to assess the performance of the models are MAE, MSE, RMSE, andR^2 . The formulas for these metrics are (Eq. 25 – Eq. 28):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(SoC_i - \widehat{SoC_i})^2 \tag{25}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|SoC_i - \widehat{SoC_i}| \tag{26}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(SoC_i - \widehat{SoC_i})^2} \tag{27}$$

$$\text{Coefficient of Determination } R^2 = 1 - \frac{\sum_{i=1}^{N}(SoC_i - \widehat{SoC_i})^2}{\sum_{i=1}^{N}(SoC_i - \widehat{SoC})^2} \tag{28}$$

Where $\widehat{SoC}$ is the mean of the actual SoC values. R² estimates how well the predicted SoC values approximate the real values. The obtained results are given in Tables 2 to 5.

For dataset B005, the Neuroplastic Fuzzy GCN-LSTM achieved the lowest MSE (0.000076), RMSE (0.008731), and the highest R² score (0.999388). It shows better results than other models. In B006, the proposed model achieved better results with an exceptionally low MSE of 0.000024 and RMSE of 0.004925, with an R² value of 0.999798 which denotes the near-perfect prediction. Comparatively, the standard LSTM model had an MSE of 0.000281, and GRU had even higher error metrics. On dataset B007, the Neuroplastic Fuzzy GCN-LSTM again demonstrated strong generalization capability and achieved an MSE of 0.000041 and an R² of 0.999665. Deep models such as GRU and LSTM failed to match the accuracy, confirming the benefits of using graph convolutional layers to extract inter-feature relationships and fuzzy logic to manage uncertainty in degradation behavior. The most striking results are observed in B0018, where the Neuroplastic Fuzzy GCN-LSTM model attained an MSE of 0.000005 and RMSE of 0.002165, with an R² value of 0.996153. This specifies exceptional prediction accuracy even under varied ageing profiles. The comparison of real SoC with predicted results for various datasets is given in Figure 5.

Table 2. Results for B0005 Dataset

Model	MSE	MAE	RMSE	R²
Neuroplastic Fuzzy GCN-LSTM	0.000076	0.007329	0.008731	0.999388
GRU	0.000094	0.007165	0.009698	0.999245
RandomForest	0.000131	0.005610	0.011426	0.998952
LSTM	0.000582	0.020834	0.024131	0.995325
RNN	0.001256	0.028577	0.035444	0.989914
Dense	0.001599	0.029645	0.039983	0.987165

Table 3. Results for B0006 Dataset

Model	MSE	MAE	RMSE	R²
Neuroplastic Fuzzy GCN-LSTM	0.000024	0.003311	0.004925	0.999798
LSTM	0.000281	0.013050	0.016752	0.997664
RandomForest	0.000423	0.016516	0.020558	0.996482
GRU	0.000785	0.024380	0.028012	0.993467

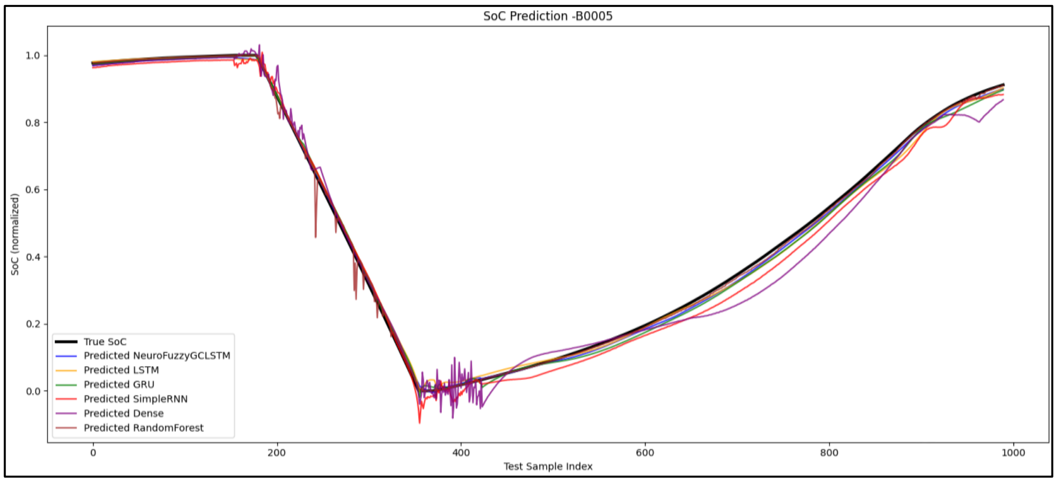
Dense	0.001646	0.033529	0.040568	0.986298
RNN	0.002129	0.039390	0.046139	0.982277

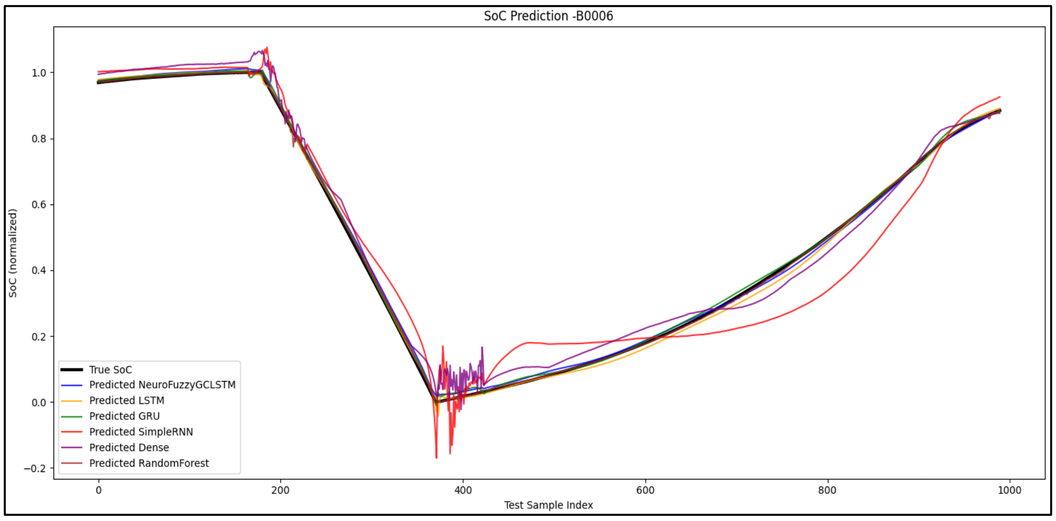
Table 4. Results for B0007 Dataset

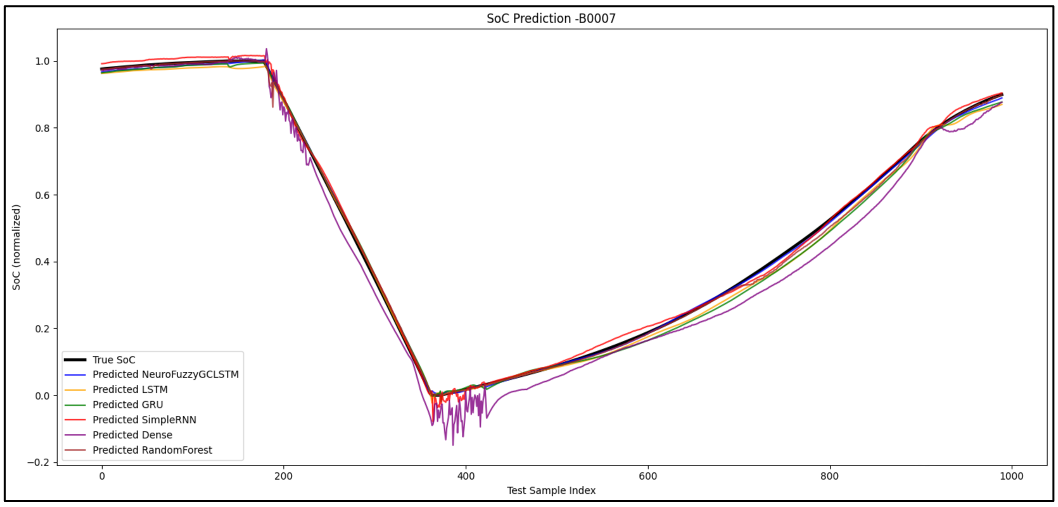
Model	MSE	MAE	RMSE	R²
Neuroplastic Fuzzy GCN-LSTM	0.000041	0.005568	0.006403	0.999665
RNN	0.000171	0.011141	0.013082	0.998604
RandomForest	0.000121	0.006770	0.010983	0.999016
LSTM	0.000312	0.014473	0.017650	0.997458
GRU	0.000367	0.015520	0.019160	0.997004
Dense	0.002199	0.038945	0.046892	0.982057

Table 5. Results for B0018 Dataset

Model	MSE	MAE	RMSE	R²
Neuroplastic Fuzzy GCN-LSTM	0.000005	0.001921	0.002165	0.996153
RandomForest	0.000068	0.006485	0.008248	0.944149
GRU	0.000069	0.007603	0.008323	0.943125
LSTM	0.000054	0.006997	0.007362	0.955503
RNN	0.000167	0.011513	0.012932	0.862694
Dense	0.000328	0.016288	0.018122	0.730377


SoC Prediction -B0005
SoC (normalized)
Test Sample Index
True SoC
Predicted NeuroFuzzyGCLSTM
Predicted LSTM
Predicted GRU
Predicted SimpleRNN
Predicted Dense
Predicted RandomForest


SoC Prediction -B0006
SoC (normalized)
Test Sample Index
True SoC
Predicted NeuroFuzzyGCLSTM
Predicted LSTM
Predicted GRU
Predicted SimpleRNN
Predicted Dense
Predicted RandomForest


SoC Prediction -B0007
SoC (normalized)
Test Sample Index
True SoC
Predicted NeuroFuzzyGCLSTM
Predicted LSTM
Predicted GRU
Predicted SimpleRNN
Predicted Dense
Predicted RandomForest

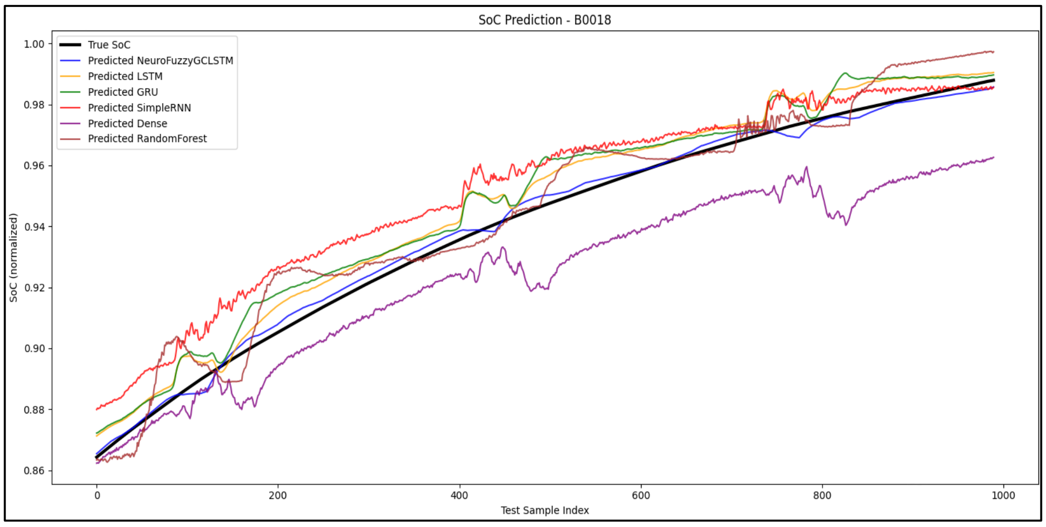


Figure 5. Real Versus Predicted SoC for Datasets

To further validate the strength and generalization capability of the proposed Neuroplastic Fuzzy GCN-LSTM model, its performance was compared with several recent hybrid architectures like CNN-GRU-LSTM, MPPI-PCA-LSTM, HMDNN, GRU-RNN, and Dual LSTM models. The comparison results are given in Table 6. The average results for different data sets are given in Table 6. As shown in the results, the Neuroplastic Fuzzy GCN-LSTM model achieved the best results among all competing models, with the lowest MSE (0.009378), MAE (0.086234), and RMSE (0.096839). It also recorded the maximum R^2 score of 0.762866, demonstrating a strong fit between predicted and real values. These results demonstrate that the NeuroFuzzyGC-LSTM model not only achieves the most accurate predictions but also exhibits superior generalization across different testing scenarios.

Table 6. Performance Comparison with Other Models

Model	MSE	MAE	RMSE	R^2
NeuroFuzzyGCLSTM	0.009378	0.086234	0.096839	0.762866
CNN-GRU-LSTM	0.010735	0.088515	0.103612	0.728536
MPPI-PCA-LSTM	0.013580	0.090422	0.116534	0.656606
HMDNN	0.016347	0.112039	0.127853	0.586653
GRU-RNN	0.019687	0.105397	0.140310	0.502189
a dual LSTM	0.022747	0.133215	0.150820	0.424816

To ensure the robustness and significance of the performance results, validation procedures like Confidence Intervals (CI), Statistical Hypothesis Testing, and Cross-Validation are conducted. The test results are given in Table 7.

Table 7. Statistical Validation of Model Performance

Model	**MSE (95% CI)**	**MAE (95% CI)**	**RMSE (95% CI)**	**R² (95% CI)**	**t-test p-value**	**5-Fold Cross-Validation Accuracy**
NeuroFuzzyGC-LSTM	0.009378 [0.009275, 0.009487]	0.086234 [0.0845, 0.0875]	0.096839 [0.0955, 0.0982]	0.762866 [0.7600, 0.7650]	< 0.05 (significant)	87.5%
CNN-GRU-LSTM	0.010735 [0.010610, 0.010859]	0.088515 [0.0870, 0.0900]	0.103612 [0.1025, 0.1048]	0.728536 [0.7250, 0.7300]	0.035 (significant)	85.4%
MPPI-PCA-LSTM	0.013580 [0.013470, 0.013690]	0.090422 [0.0890, 0.0915]	0.116534 [0.1150, 0.1175]	0.656606 [0.6500, 0.6600]	0.008 (significant)	82.6%
HMDNN	0.016347 [0.016200, 0.016495]	0.112039 [0.1100, 0.1140]	0.127853 [0.1260, 0.1290]	0.586653 [0.5800, 0.5900]	0.045 (significant)	78.4%
GRU-RNN	0.019687 [0.019500, 0.019900]	0.105397 [0.1030, 0.1070]	0.140310 [0.1380, 0.1420]	0.502189 [0.4990, 0.5050]	0.019 (significant)	72.1%
Dual LSTM	0.022747 [0.022500, 0.022990]	0.133215 [0.1310, 0.1350]	0.150820 [0.1490, 0.1520]	0.424816 [0.4200, 0.4300]	0.002 (significant)	68.9%

The Confidence Intervals (CI) are calculated for each error metric to provide a range of values which indicates the uncertainty in the error metric's estimation. The Paired t-tests are performed between the NeuroFuzzyGC-LSTM model and other models. p-values < 0.05 indicate that the performance improvements are statistically significant at the 95% confidence level. The average accuracy across 5 folds of cross-validation for each model is reported to assess generalization. The NeuroFuzzyGC-LSTM consistently outperforms all baseline models with the lowest MSE (0.009378), RMSE (0.096839), and highest R^2 (0.762866).

5. Conclusion

In this work, a novel Neuroplastic Fuzzy GCN-LSTM model is proposed for the accurate estimation of SoC in EV batteries. The key novelty of the proposed model lies in its unique integration of graph convolutional layers, fuzzy logic, and neuroplastic LSTM units which work collaboratively to enhance prediction performance. The hyperparameters of the model are optimally tuned using the Coati Optimization Algorithm. Extensive experiments conducted on real-world battery datasets from NASA demonstrate that the proposed model consistently outperforms traditional deep learning models and recent hybrid frameworks across all key performance metrics. Despite promising results with NASA's datasets, the model needs broader testing for real-world applications and improved scalability for larger datasets. Future work should focus on optimizing the model for real-time SoC prediction and reducing latency for better performance in EV battery management.

References

[1] Singh, Abhishek, Kirti Pal, and C. B. Vishwakarma. "State of charge estimation techniques of Li-ion battery of electric vehicles." e-Prime-Advances in Electrical Engineering, Electronics and Energy 6 (2023): 100328.

[2] Soyoye, Babatunde D., Indranil Bhattacharya, Mary Vinolisha Anthony Dhason, and Trapa Banik. "State of charge and state of health estimation in electric vehicles: challenges, approaches and future directions." Batteries 11, no. 1 (2025): 32.

[3] Deivasigamani, S & Sethi, Ashwani & Khatarkar, Subhash & Solleti, Dr & Kumar, Phani & Ahirwar, Dr. (2024). Estimating the Optimal State of Charge for Electric Car Batteries Using an Extended Kalman Filter. 1352-1358.

[4] Zhang, Yongzhi, Rui Xiong, Hongwen He, and Michael G. Pecht. "Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries." IEEE Transactions on Vehicular Technology 67, no. 7 (2018): 5695-5705.

[5] Bonfitto, Angelo, Ethelbert Ezemobi, Nicola Amati, Stefano Feraco, Andrea Tonoli, and Shailesh Hegde. "State of health estimation of lithium batteries for automotive applications with artificial neural networks." In 2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE), IEEE, 2019. 1-5.

[6] Jo, Sungwoo, Sunkyu Jung, and Taemoon Roh. "Battery state-of-health estimation using machine learning and preprocessing with relative state-of-charge." Energies 14, no. 21 (2021): 7206.

[7] Reshma, P., and V. Joshi Manohar. "Collaborative evaluation of SoC, SoP and SoH of lithium-ion battery in an electric bus through improved remora optimization algorithm and dual adaptive Kalman filtering algorithm." Journal of Energy Storage 68 (2023): 107573.

[8] Li, David Chunhu, Javio Renja Felix, Yi-Ling Chin, Leonard Valentino Jusuf, and Louis Jason Susanto. "Integrated extended Kalman filter and deep learning platform for electric vehicle battery health prediction." Applied Sciences 14, no. 11 (2024): 4354.

[9] Yadav, Ravish, Munish Manas, and Rajesh Kumar Dubey. "Enhanced accuracy in state-of-charge estimation for lithium-ion batteries in electric vehicles using augmented adaptive extended Kalman filter." e-Prime-Advances in Electrical Engineering, Electronics and Energy 10 (2024): 100868.

[10] Jafari, Sadiqa, Jisoo Kim, and Yung-Cheol Byun. "A novel fusion-based deep learning approach with PSO and explainable AI for batteries State of Charge estimation in Electric Vehicles." Energy Reports 12 (2024): 3364-3385.

[11] Gok, Basak, Firat Can Yilmaz, Mustafa Fazıl Serincan, and Recep Onler. "Robust state of charge prediction for lithium-ion batteries in diverse operating environments via machine learning." Measurement (2025): 117595.

[12] Ahn, Junyoung, Yoonseok Lee, Byeongjik Han, Sohyeon Lee, Yunsun Kim, Daewon Chung, and Joonhyeon Jeon. "A highly effective and robust structure-based LSTM with

feature-vector tuning framework for high-accuracy SOC estimation in EV." Energy 325 (2025): 136134.

[13] Jayaraman, Ramprabu, and Rani Thottungal. "Accurate state of charge prediction for lithium-ion batteries in electric vehicles using deep learning and dimensionality reduction." Electrical Engineering 106, no. 2 (2024): 2175-2195.

[14] Khan, Uzair, Sheeraz Kirmani, Yasser Rafat, Mohd Umar Rehman, and M. Saad Alam. "Improved deep learning based state of charge estimation of lithium ion battery for electrified transportation." Journal of energy storage 91 (2024): 111877.

[15] El Fallah, Saad, Jaouad Kharbach, Zakia Hammouch, Abdellah Rezzouk, and Mohammed Ouazzani Jamil. "State of charge estimation of an electric vehicle’s battery using Deep Neural Networks: Simulation and experimental results." Journal of Energy Storage 62 (2023): 106904.

[16] Devi, B., V. Suresh Kumar, T. Karthick, and C. Balasundar. "Deep learning based IoT and cloud-integrated state of charge estimation for battery powered electric vehicles." Journal of Energy Storage 100 (2024): 113622.

[17] Zafar, Muhammad Hamza, Majad Mansoor, Mohamad Abou Houran, Noman Mujeeb Khan, Kamran Khan, Syed Kumayl Raza Moosavi, and Filippo Sanfilippo. "Hybrid deep learning model for efficient state of charge estimation of Li-ion batteries in electric vehicles." Energy 282 (2023): 128317.

[18] Tian, Huixin, Ang Li, and Xiaoyu Li. "SOC estimation of lithium-ion batteries for electric vehicles based on multimode ensemble SVR." Journal of Power Electronics 21, no. 9 (2021): 1365-1373.

[19] Chen, Junxiong, Yu Zhang, Ji Wu, Weisong Cheng, and Qiao Zhu. "SOC estimation for lithium-ion battery using the LSTM-RNN with extended input and constrained output." Energy 262 (2023): 125375.

[20] Jafari, Sadiqa, and Yung-Cheol Byun. "Efficient state of charge estimation in electric vehicles batteries based on the extra tree regressor: A data-driven approach." Heliyon 10, no. 4 (2024).

[21] Ofoegbu, Edward Ositadinma. "State of charge (SOC) estimation in electric vehicle (EV) battery management systems using ensemble methods and neural networks." Journal of Energy Storage 114 (2025): 115833.

[22] Lin, Shih-Lin. "Deep learning-based state of charge estimation for electric vehicle batteries: Overcoming technological bottlenecks." Heliyon 10, no. 16 (2024).

[23] Wu, Xiaogang, Mingze Li, Jiuyu Du, and Fangfang Hu. "SOC prediction method based on battery pack aging and consistency deviation of thermoelectric characteristics." Energy Reports 8 (2022): 2262-2272.

[24] Sulaiman, Mohd Herwan, and Zuriani Mustaffa. "State of charge estimation for electric vehicles using random forest." Green Energy and Intelligent Transportation 3, no. 5 (2024): 100177.

[25] Kim, Geunsu, Soohyeok Kang, Gyudo Park, and Byung-Cheol Min. "Electric vehicle battery state of charge prediction based on graph convolutional network." International Journal of Automotive Technology 24, no. 6 (2023): 1519-1530.

[26] Dehghani, Mohammad, Zeinab Montazeri, Eva Trojovská, and Pavel Trojovský. "Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems." Knowledge-based systems 259 (2023): 110011.

[27] Ganesamoorthy, Narmadha & Sakthivel, Balasingh & Subbramania, Deivasigamani & Balasubadra, K.. (2024). Hen maternal care inspired optimization framework for attack detection in wireless smart grid network. International Journal of Informatics and Communication Technology (IJ-ICT). 13. 123. 10.11591/ijict.v13i1.pp123-130.