

A Voltage-Controlled Model Machine Learning Framework for State-of-Charge Estimation in Battery Management System: A Performance Evaluation

Ignatius Agus Supriyono^{1,*}, Eko Sedyono², Iwan Setyawan³, Ivanna Kristianti Timotius⁴,
Untung Rahardja⁵

^{1,2,3,4}Satya Wacana Christian University, Salatiga, Central Java, Indonesia

⁵Universitas Raharja, Tangerang, Banten, Indonesia

(Received: April 1, 2026; Revised: June 1, 2026; Accepted: September 10, 2026; Available online: October 14, 2026)

Abstract

Accurate State-of-Charge (SoC) estimation is essential for ensuring the reliability, safety, and efficient operation of Battery Energy Storage Systems (BESS), particularly in photovoltaic (PV) applications characterized by intermittent generation. Conventional SoC estimation methods may have limitations in capturing nonlinear and time-dependent voltage behavior under practical operating conditions. This study proposes a data-driven SoC estimation framework that integrates a Voltage-Controlled Model (VCM) with sliding-window temporal feature extraction and multiple machine learning (ML) algorithms. The sliding-window approach transforms sequential voltage measurements into structured lag-based features to represent battery discharge dynamics more effectively. Real-world voltage data were collected from four 45-Ah lead-acid batteries operating in a PV-BESS system from February to December 2024. More than 140,000 charge-discharge records were acquired, from which 6,226 discharge records were extracted for model development and evaluation. Five ML models, namely Support Vector Regression (SVR), Artificial Neural Network (ANN), Random Forest (RF), Long Short-Term Memory (LSTM), and eXtreme Gradient Boosting (XGBoost), were compared using mean squared error (MSE), mean absolute error (MAE), root mean squared error (RMSE), and computation time. The results show that VCM-LSTM achieves the lowest prediction errors, whereas VCM-XGBoost provides the most favorable balance between predictive accuracy and computational efficiency. VCM-XGBoost achieves MSE = 0.0849, MAE = 0.1401, RMSE = 0.2913, and a computation time of 0.94 seconds. These findings indicate that combining sliding-window temporal representation with VCM and ML can support accurate and computationally efficient SoC estimation, highlighting the potential of VCM-XGBoost for resource-constrained BMS applications, particularly in real-world residential PV-BESS environments with limited computational resources.

Keywords: State-of-Charge, Voltage-Controlled Model, Sliding Window, Machine Learning, Battery Management System, Performance Evaluation

1. Introduction

The rapid growth of photovoltaic (PV) renewable energy presents a promising pathway toward sustainable electricity generation [1], [2], while simultaneously introducing operational challenges due to the inherently intermittent and uncertain nature of solar resources [3], [4]. In this context, battery management systems (BMS) play a critical role in maintaining system reliability and ensuring continuous energy availability [5], [6]. Prior studies have demonstrated the feasibility of real-time State-of-Charge (SoC) acquisition using embedded platforms and intelligent control techniques, such as stochastic optimal control for battery applications [7], [8]. In parallel, advances in dynamic energy scheduling have highlighted the importance of algorithm-based strategies to improve energy utilization under variable operating conditions [9], [10]. Within battery energy storage systems (BESS), accurate SoC estimation remains essential for optimal energy management and system efficiency [11].

Extensive research on SoC estimation particularly in lithium-ion batteries covers both direct methods, such as ampere-hour counting and open-circuit voltage mapping, and indirect methods, including kalman filter variants and sliding mode observers [12], [13]. Although effective in controlled environments, these approaches often struggle to address

*Corresponding author: Ignatius Agus Supriyono ((982022014@student.uksw.edu)

DOI: <https://doi.org/10.47738/jads.v7i4.1511>

This is an open access article under the CC-BY license (<https://creativecommons.org/licenses/by/4.0/>).

© Authors retain all copyrights

nonlinear voltage behavior, fluctuating load conditions, and long-term degradation effects encountered in real-world deployments. To overcome these limitations, recent studies have increasingly explored data-driven and Machine Learning (ML) techniques capable of capturing nonlinear and time-varying battery dynamics, thereby improving estimation accuracy and operational adaptability [14]. However, existing studies have not sufficiently explored sliding window-based temporal feature engineering for improving both accuracy and computational efficiency in SoC estimation.

Building on these developments, this study introduces a voltage-controlled model (VCM) and machine learning (ML) framework to enhance SoC estimation from time-series battery data. The sliding window approach transforms sequential voltage measurements into structured temporal features, enabling the model to capture discharge patterns and dynamic transitions more effectively. This strategy supports more reliable SoC estimation, which is fundamental for ensuring safe operation, extending battery lifetime, and improving energy utilization in PV-based systems [15], [16].

To further strengthen the modeling capability, this work adopts the VCM as a physics-guided representation of the nonlinear relationship between battery voltage and SoC [17]. By integrating VCM with sliding window feature extraction, the proposed framework combines physical insight with data-driven learning to enhance predictive performance [18], [19], [20]. The resulting hybrid approach leverages temporal voltage trajectories to improve model accuracy while maintaining computational efficiency.

2. Literature Review

The BMS plays a critical role in renewable energy systems that rely on multiple batteries for energy storage, where accurate SoC estimation is essential to ensure reliable operation and prevent deep discharge or overcharging. Estimation techniques are generally classified into model-based and data-driven approaches. Model-based methods, such as open-circuit voltage (OCV) estimation [21], Adaptive Kalman Filter (AKF) [22], and other Kalman filter variants have been widely investigated. Although these approaches are mathematically well-established, they typically require high computational effort and struggle to fully capture the nonlinear and time-varying characteristics of battery behavior, limiting their suitability for real-time BMS applications.

To address these limitations, recent studies have explored ML-based methods, which are more capable of modeling nonlinear dynamics from data. Early work includes the integration of LSTM-RNN [23], as well as least square SVM (LSSVM) [24], alongside optimization-driven methods such as GRU-LSTM [25], and deep learning architectures like Convolutional Neural Network (CNN) with U-Net architecture [26], [27], [28], these approaches require high-quality datasets and often involve significant computational complexity.

Recent research trends emphasize data-driven time-series frameworks for SoC estimation, particularly for large-scale and real-time applications. Reviews in [29], [30], [31] indicate that ML-based approaches can effectively handle nonlinear and dynamic battery characteristics, although many studies do not report computational efficiency.

J. A. Braun et al. [32] introduced VCM. They provide a promising alternative by exploiting the relationship between battery voltage and SoC. This approach enables SoC estimation using voltage-only measurements and leverages temporal patterns, particularly sharp voltage drops as indicators of low charge states [33]. Complementing this finding, ML techniques such as Artificial Neural Networks (ANN) [34], Support Vector Machines (SVM) [35], Random Forest (RF) [36], Long Short-Term Memory (LSTM) [37], and eXtreme Gradient Boosting (XGBoost) [38], [39] have demonstrated strong capability in capturing nonlinear battery behavior [40], [41].

Building on these observations, this work adopts a VCM-ML framework to enhance SoC estimation. The sliding window technique transforms raw voltage time-series into lag-based feature representations, enabling models to capture temporal discharge patterns more effectively. By integrating this time-series feature extraction with ML algorithms, the proposed approach improves the ability to model nonlinear battery dynamics while maintaining computational efficiency. Furthermore, the combination of sliding window representation with the VCM provides a hybrid data-driven framework that leverages both physical insight and statistical learning.

Motivated by the need for practical BMS deployment, this research aims to evaluate the performance of multiple regression models, including SVR, ANN, RF, LSTM, and XGBoost, using real-world PV-based lead-acid battery data. The contributions of this work include: (i) the development of a real-world experimental platform with cloud-based data acquisition, (ii) the application of sliding window time-series feature engineering for SoC estimation based on VCM, (iii) a comparative evaluation of multiple ML models in terms of accuracy and computational efficiency, and (iv) the validation of the proposed framework under realistic residential PV-BESS operating conditions.

3. Methodology

3.1. Study Design

This research adopts an experimental approach to evaluate the SoC of lead-acid batteries in a PV energy system. Real-time voltage measurement is acquired using a microcontroller connected to the internet and transmitted to a cloud database. The collected data is subsequently downloaded and validated using Python, with NumPy and Pandas for preprocessing and quality checks, and Matplotlib and Seaborn for visualization. Following validation, the dataset is used to develop machine learning models using Scikit-learn, XGBoost, and TensorFlow for SoC estimation. To provide a more straightforward overview of the proposed methodology, the research workflow is illustrated in [figure 1](#).

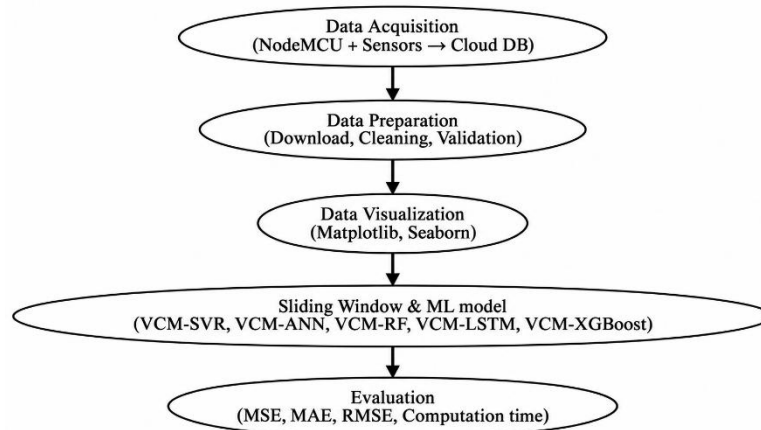


Figure 1. Research workflow

Figure 1 illustrates the process begins with real-time data acquisition from lead-acid batteries using NodeMCU microcontrollers equipped with sensors, followed by data preparation to ensure the quality of the dataset. The processed data is then visualized to observe discharge patterns before being used in machine learning models for SoC estimation [42]. Finally, the models are evaluated based on accuracy and computation time to determine the most efficient approach. The VCM-SVR, VCM-ANN, VCM-RF, VCM-LSTM, and VCM-XGBoost are used in this work.

3.2. Experiment Setup

This experiment utilized four pieces of 100WP monocrystalline PV panels, four lead-acid batteries with a capacity (C) of 45Ah were employed, along with four 50A MPPT charge controllers and Node MCU ESP8266 units equipped with ACS758 voltage sensors to send battery voltage data to the cloud database. The household electrical load consisted of several lighting fixtures, with rated at 24 watts, operating for 12 hours at night. One experimental setup is shown in [figure 2\(a\)](#), and the household electrical load is shown in [figure 2\(b\)](#). There are four experiment benches, but only one set is shown, as the others are similar. [Figure 2\(a\)](#) illustrates the components on the experimental table, which comprise a battery (1), microcontroller (2), voltage sensor (3), current sensor (4), charger controller (5), and inverter (6). [Figure 2\(b\)](#) shows the use of electrical power from batteries for household lighting.

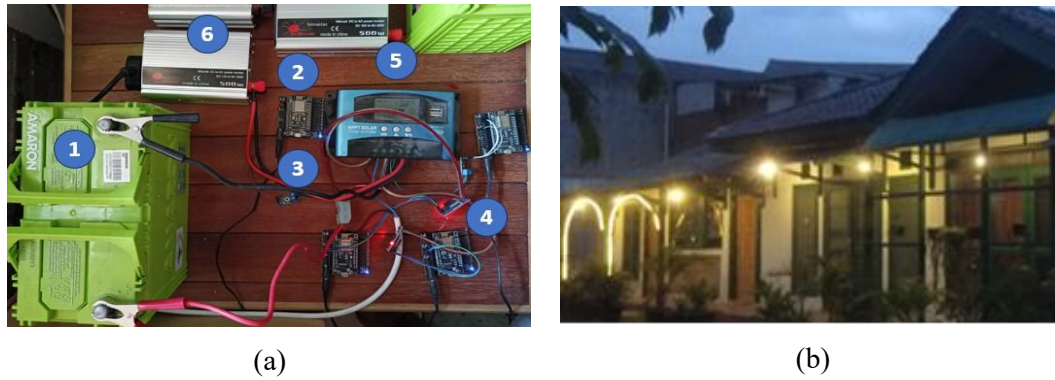


Figure 2. Experiment bench (a) and household electric load (b)

3.3. Data Acquisition

Data acquisition was conducted by installing NodeMCU ESP8266 equipped with voltage sensor, then sending real-time voltage from the battery every ten minutes. The voltage data of the battery were stored in a MySQL database on the cloud [29]. In this data acquisition process, JavaScript was used for communication between the NodeMCU ESP8266 and the MySQL database. Additionally, a PHP script was used to display real-time data for over 12 hours on a dashboard, with a button to download the data from the cloud to a local computer. The data acquisition device is shown in figure 2(a), and the framework is illustrated in figure 3. This article used only voltage data without utilizing current data to predict the SoC.



Figure 3. Data acquisition framework

Figure 3 illustrates the data acquisition framework, in which each component plays a distinct functional role. The energy sources supply electrical power, which is managed and stored by the energy storage system [43], [44], [45]. Household electrical loads represent the demand side of the system, while user monitoring devices enable real-time observation of system performance. Cloud-based application software processes and analyzes the collected data, which is then stored in a centralized cloud database for long-term accessibility. A graphical dashboard interface provides users with a visual representation of the system's status, and the load switching control mechanism ensures the efficient allocation and continuity of power delivery.

3.4. Data Preparation

The data are stored in the cloud, downloaded in Excel format, and then converted to a CSV file. Subsequently, it is validated and processed in Python using the NumPy and Pandas libraries to ensure the datasets are suitable and ready for use. The data acquisition and modeling are illustrated in figure 4. Figure 4 illustrates the four battery voltages over the ten months from February to December 2024, comprising approximately 35,000 records each battery. The data in the cloud database were downloaded in Excel (.XLS) format, and then converted to a Comma-Separated Value (.CSV) file. Subsequently, the data are validated and processed in Python using the NumPy and Pandas libraries to ensure it is suitable and ready for use ML modeling.

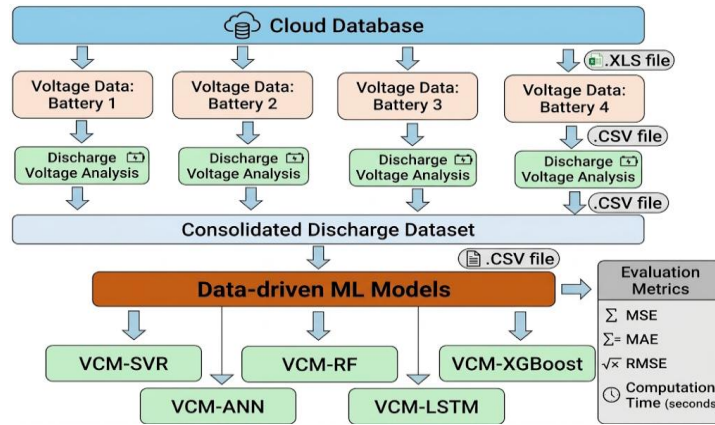


Figure 4. Data preparation illustration

The discharge data is split from charge discharge data is shown in figure 5. Figure 5 shows the discharge dataset which is collected from February 2024 to December 2024. It consists of 6,226 records.

...	Time (t)	Voltage (v)	Low SoC threshold	Battery	Date-Time
0	1	12.10	10	Battery 1	02/11/2024 01:34:00
1	2	10.17	10	Battery 1	02/11/2024 01:43:00
2	3	10.14	10	Battery 1	02/11/2024 01:54:00
3	4	10.00	10	Battery 1	02/11/2024 02:04:00
4	5	9.90	10	Battery 1	02/11/2024 02:14:00
...	...	...	...	...	...
6221	6222	10.55	10	Battery 4	12/30/2024 02:50:00
6222	6223	10.53	10	Battery 4	12/30/2024 03:00:00
6223	6224	10.46	10	Battery 4	12/30/2024 03:10:00
6224	6225	10.31	10	Battery 4	12/30/2024 03:20:00
6225	6226	9.87	10	Battery 4	12/30/2024 03:30:00

[6226 rows x 5 columns]

Figure 5. Discharge dataset

3.5. Predictive Modeling

In this study, sliding window technique is used as an input for estimation model. Based on the experiment, the optimal window size is 36, indicating that the optimal period from full capacity (C-full) to low capacity (C-low) is 36 time-steps. This value is generated from data delivered every 10 minutes, so 36 samples are equivalent to 6 hours. ML algorithms estimate SoC using datasets downloaded from a cloud database. The algorithms used in this research are SVR, ANN, RF, LSTM, and XGBoost. The equations of sliding window, SVR, ANN, RF, LSTM, XGBoost are shown in equations (1-11).

3.6. Sliding Window

A sliding window is a dynamic technique for analyzing sequential data by continuously shifting the window of observations. In this case, the window size is set to 36, meaning the system always considers the most recent 36 samples equivalent to six hours of data collected every ten minutes. As new data arrives, the window “slides” forward, dropping the oldest point and adding the newest, ensuring that the analysis always reflects the latest six-hour period without losing historical context. To capture temporal dependencies in battery behavior, a sliding window technique is applied to the real-time voltage signal. The selection of 36 lag steps is based on the observed discharge cycle duration, providing sufficient temporal coverage for capturing battery behavior dynamics. Let the voltage sequence be:

$$V = \{v_1, v_2, v_3, \dots, v_t\} \quad (1)$$

v_t denotes the voltage measurement at time step t , and T represents the total number of observations. A sliding window with a length of $L = 36$ is defined as

$$X_t = [v_{t-35}, v_{t-34}, \dots, v_t] \quad (2)$$

X_t represents the input feature vector at time step t , and $L = 36$ denotes the number of historical voltage observations included in each window.

Each window captures short-term voltage variations that characterize the battery discharge process. The resulting feature vectors are paired with their corresponding target SoC values to construct a supervised learning dataset:

$$\mathcal{D} = \{(X_t, y_t)\}_{t=L}^T \quad (3)$$

y_t denotes the target SoC value associated with the input window X_t . This transformation converts the original voltage time series into structured input features suitable for regression models. By incorporating historical voltage measurements, the sliding window technique enables machine learning algorithms to learn temporal dependencies and nonlinear patterns associated with battery discharge behavior.

3.7. Support Vector Regression

Support Vector Regression (SVR) is a regression technique that estimates continuous target values by identifying a function that minimizes model complexity while allowing prediction errors within a predefined tolerance ε . In this study, SVR is applied to estimate battery SoC using the 36-dimensional voltage feature vectors generated by the sliding window technique. The SVR prediction function is expressed as

$$f(X_t) = \mathbf{w}^T \phi(X_t) + b \quad (4)$$

X_t represents the input voltage feature vector, $\mathbf{w}$ denotes the weight vector, $\phi(\cdot)$ is a nonlinear feature mapping function, and b represents the bias term. The SVR model employs a kernel function to evaluate relationships between input observations in the transformed feature space. A radial basis function (RBF) kernel is one example of a kernel that can capture nonlinear relationships between voltage measurements and SoC. The model parameters are estimated by minimizing prediction errors while controlling the complexity of the regression function. The use of SVR allows the framework to model nonlinear relationships between historical voltage patterns and battery SoC while maintaining a structured regression formulation.

3.8. Artificial Neural Network

An Artificial Neural Network (ANN) is a machine learning model composed of interconnected neurons organized into layers. Each neuron processes its input through weighted connections, bias terms, and an activation function. In this study, ANN is employed to estimate battery SoC from the 36-dimensional voltage feature vectors obtained through sliding window feature extraction. A typical ANN with one hidden layer can be formulated as

$$\hat{y}_t = g\left(\sum_{j=1}^H w_j^{(2)} h\left(\sum_{i=1}^L w_{ji}^{(1)} v_{t-L+i} + b_j^{(1)}\right) + b^{(2)}\right) \quad (5)$$

With $L = 36$ denotes the number of input features, H represents the number of hidden neurons, $w_{ji}^{(1)}$ denotes the weight connecting input neuron i to hidden neuron j , and $w_j^{(2)}$ represents the weight connecting hidden neuron j to the output layer. The terms $b_j^{(1)}$ and $b^{(2)}$ denote the hidden-layer and output-layer biases, respectively. The functions $h(\cdot)$ and $g(\cdot)$ represent the hidden-layer and output-layer activation functions. Through training, the network adjusts its weights and biases to minimize the difference between predicted and target SoC values. The ANN model can therefore learn nonlinear relationships between voltage patterns and battery charge conditions.

3.9. Random Forest

Random Forest (RF) is an ensemble learning method that combines multiple decision trees to generate predictions. For regression tasks, the final prediction is obtained by averaging the outputs of the individual trees. In this study, RF is used to estimate battery SoC from the 36-dimensional voltage feature vectors. The RF regression prediction is formulated as

$$\hat{y}_t = \frac{1}{N} \sum_{k=1}^N f_k(X_t) \quad (6)$$

N represents the number of decision trees, X_t denotes the input voltage feature vector, and $f_k(X_t)$ is the prediction generated by the k -th regression tree. Each tree is constructed using the training data and a randomized selection of samples and features. The ensemble combines the predictions of multiple trees to produce the final SoC estimate. By aggregating individual tree outputs, RF can capture nonlinear relationships between voltage measurements and SoC without requiring an explicitly defined mathematical relationship between the input and target variables.

3.10. Long Short-Term Memory

Long Short-Term Memory (LSTM) is a type of recurrent neural network designed to learn temporal dependencies in sequential data. Unlike conventional feedforward networks, LSTM incorporates memory cells and gating mechanisms that regulate the flow of information across time steps. In this study, LSTM processes a sequence of 36 historical battery voltage measurements to estimate SoC. At each time step i , the LSTM cell updates its internal states through the following operations:

$$\begin{aligned} f_i &= \sigma(W_f[v_i, h_{i-1}] + b_f) \\ i_i &= \sigma(W_i[v_i, h_{i-1}] + b_i) \\ \tilde{C}_i &= \tanh(W_c[v_i, h_{i-1}] + b_c) \\ C_i &= f_i \odot C_{i-1} + i_i \odot \tilde{C}_i \\ o_i &= \sigma(W_o[v_i, h_{i-1}] + b_o) \\ h_i &= o_i \odot \tanh(C_i) \end{aligned} \quad (7)$$

v_i denotes the input voltage at time step i , h_{i-1} represents the previous hidden state, and C_{i-1} denotes the previous cell state. The variables f_i , i_i , and o_i represent the forget, input, and output gates, respectively. The candidate cell state is represented by $\tilde{C}_i$, while W and b denote the corresponding weight matrices and bias vectors. The function $\sigma(\cdot)$ is the sigmoid activation function, and $\odot$ denotes element-wise multiplication. The final SoC prediction is obtained from the last hidden state:

$$\hat{y}_t = W_y h_L + b_y \quad (8)$$

With $L = 36$ represents the sequence length, W_y is the output weight matrix, and b_y is the output bias. By maintaining information across the input sequence, LSTM can learn temporal patterns associated with battery discharge and capture changes in voltage behavior that may be relevant to SoC estimation.

3.11. Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is an ensemble learning algorithm that builds regression trees sequentially, with each successive tree improving the predictions generated by the existing ensemble. XGBoost incorporates regularization into its objective function to control model complexity and reduce overfitting. Its computationally efficient implementation also supports parallel processing during tree construction, making it suitable for nonlinear regression tasks. In this study, XGBoost estimates battery SoC using the 36-dimensional voltage feature vectors obtained from the sliding window technique. The prediction is calculated as the sum of the outputs from the individual regression trees:

$$\hat{y}_t = \sum_{k=1}^K f_k(X_t), \quad f_k \in \mathcal{F} \quad (9)$$

K denotes the number of regression trees, X_t represents the input feature vector, f_k denotes the prediction function of the k -th tree, and $\mathcal{F}$ represents the space of regression trees. The model is trained by minimizing a regularized objective function:

$$\mathcal{L} = \sum_{t=1}^M l(y_t, \hat{y}_t) + \sum_{k=1}^K \Omega(f_k) \quad (10)$$

M denotes the number of training observations, $l(\cdot)$ represents the loss function measuring the difference between actual and predicted SoC values, and $\Omega(f_k)$ represents the regularization term associated with the k -th tree. The regularization term can be expressed as

$$\Omega(f) = \gamma T_f + \frac{1}{2} \lambda \sum_{j=1}^{T_f} w_j^2 \quad (11)$$

T_f denotes the number of leaves in a tree, w_j represents the score assigned to leaf j , and γ and λ are regularization parameters that control tree complexity and leaf-weight magnitude. By combining sequential boosting with regularized optimization, XGBoost can model nonlinear relationships between historical voltage measurements and SoC while maintaining computational efficiency.

3.12. Performance Evaluation

The performance of the evaluated machine learning models is assessed using three error metrics: Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). Computational efficiency is evaluated by recording the execution time required by each model in the Google Colab environment. MAE measures the average absolute difference between actual and predicted values and is defined as

$$\text{MAE} = \frac{1}{M} \sum_{i=1}^M |y_i - \hat{y}_i| \quad (12)$$

MSE measures the average squared prediction error and is expressed as

$$\text{MSE} = \frac{1}{M} \sum_{i=1}^M (y_i - \hat{y}_i)^2 \quad (13)$$

RMSE is calculated as the square root of MSE and is defined as

$$\text{RMSE} = \sqrt{\frac{1}{M} \sum_{i=1}^M (y_i - \hat{y}_i)^2} \quad (14)$$

M represents the number of evaluated samples, y_i denotes the actual target value, and $\hat{y}_i$ represents the predicted value. Lower MAE, MSE, and RMSE values indicate smaller prediction errors. Execution time is reported separately to assess the computational efficiency of each model. Considering both prediction accuracy and computational cost provides a basis for comparing the suitability of the evaluated algorithms for battery management applications.

3.13. Hyperparameter Optimization

Hyperparameter optimization was conducted to establish a systematic comparison of machine learning models and identify an appropriate configuration for the XGBoost model before final evaluation [46], [47]. Because the battery voltage measurements constitute sequential time-series data, temporal cross-validation was employed to preserve the chronological order of observations during model selection. The XGBoost optimization process considered three principal hyperparameters: the number of estimators, maximum tree depth, and learning rate. The number of estimators was evaluated using values of 100, 200, 300, and 400. Maximum tree depth was examined at values of 3, 5, and 7, while the learning rate was evaluated at 0.05, 0.10, 0.20, and 0.30.

The optimization procedure was performed exclusively on the training dataset. Temporal cross-validation was used to compare candidate configurations while maintaining the temporal structure of the observations. The configuration producing the lowest validation error was selected as the optimized XGBoost model. The held-out test dataset was reserved for the final performance evaluation to avoid using test observations during hyperparameter selection.

3.14. Computational Environment and Tools

The experiments were conducted using the Google Colab computing environment. Data processing and manipulation were performed using Pandas and NumPy, while TensorFlow and Keras were used for neural network development. XGBoost was employed to implement the gradient-boosting model, and Scikit-learn was used for machine learning model development and evaluation. Matplotlib and Seaborn were utilized to visualize the voltage data and model evaluation results. The reported hardware configuration comprised an AMD Ryzen 7 4700U processor operating at 2.00 GHz, AMD Radeon Graphics with 2 GB of graphics memory, 8 GB of RAM, and a 64-bit operating system running on an x64-based processor. These tools and computing resources supported the data preparation, model training, performance evaluation, and visualization procedures conducted in this study.

4. Results and Discussion

4.1. Data Visualization

The voltage dataset, which is illustrated in figure 5, can be visualized using the Matplotlib and Seaborn libraries for further analysis. The visualization of Battery-1 and Battery-2 data is presented in figure 6 and figure 7. Only Battery-1 and Battery-2 are presented, as Battery-3 and Battery-4 have identical characteristics.

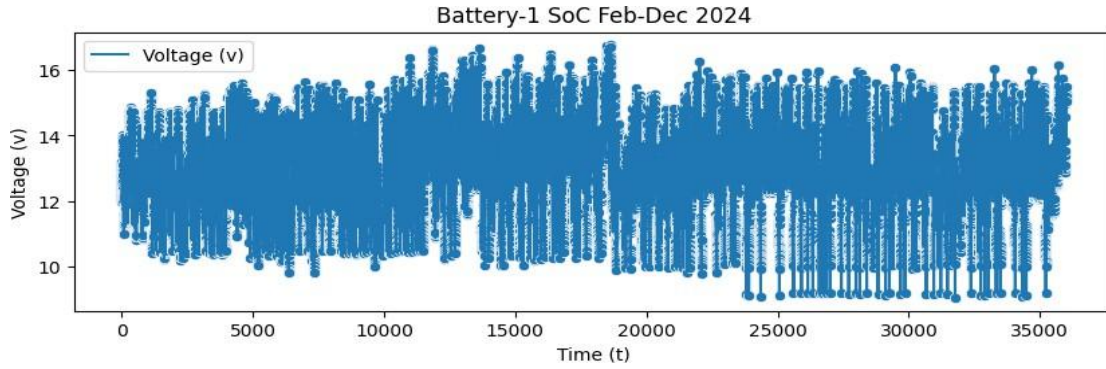


Figure 6. Battery-1 voltage data visualization

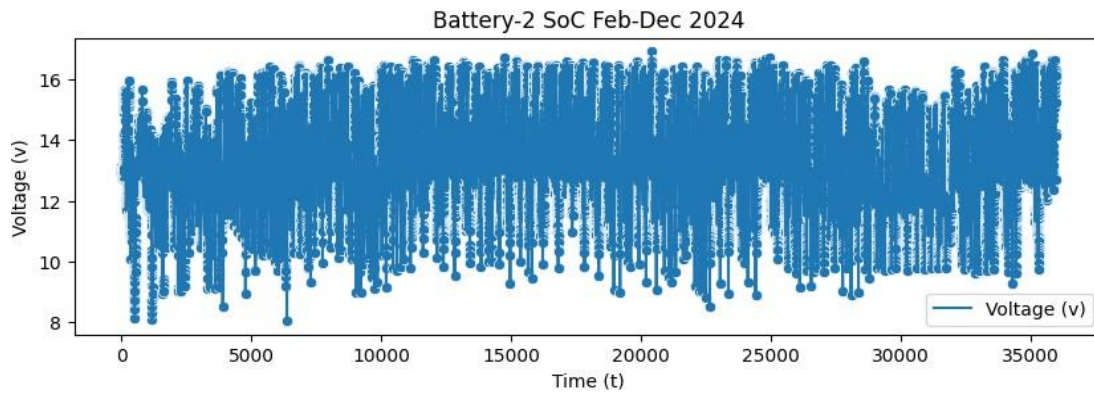


Figure 7. Battery-2 voltage data visualization.

In this work, from the total 140,000 records of charge and discharge data across four batteries, only the discharge data were extracted, resulting 6,226 records is shown in [figure 8](#). The discharge phase was selected because it directly represents the battery's ability to supply energy to the load. [Figure 8](#) shows the visualization of the battery discharge voltage from February to December 2024. Unlike the charging process, discharge data are more relevant for identifying when the SoC approaches the low threshold. This makes discharge data crucial for accurate SoC estimation and timely switching to another battery with sufficient energy. In this discharge dataset, we add a column called “Low SoC Threshold” with a value of 10 volts to the discharge dataset, based on the experiment where 10 volts indicates that the battery is approaching a low SoC and should stop supplying energy to avoid deep discharge. Based on observation in this experiment, the battery discharge voltage is between 9 to 13 volts, in case battery discharge operation under 9 volts, the charge controller (MPPT) will physically cut-off the load. Therefore 10 volts of discharge range 9 to 13 volts means SoC of 25%.

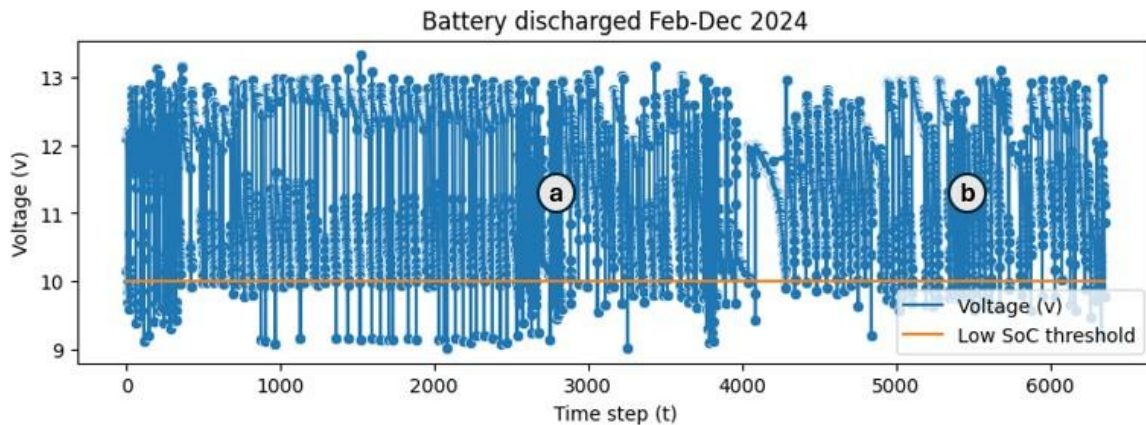


Figure 8. Battery discharge visualization

4.2. Voltage-Controlled Model for SoC

In the experiments presented above, it can be demonstrated that utilizing microcontrollers, as mentioned in [32] enables the measurement of voltage as shown in figure 9 for further data processing using ML to predict the SoC.

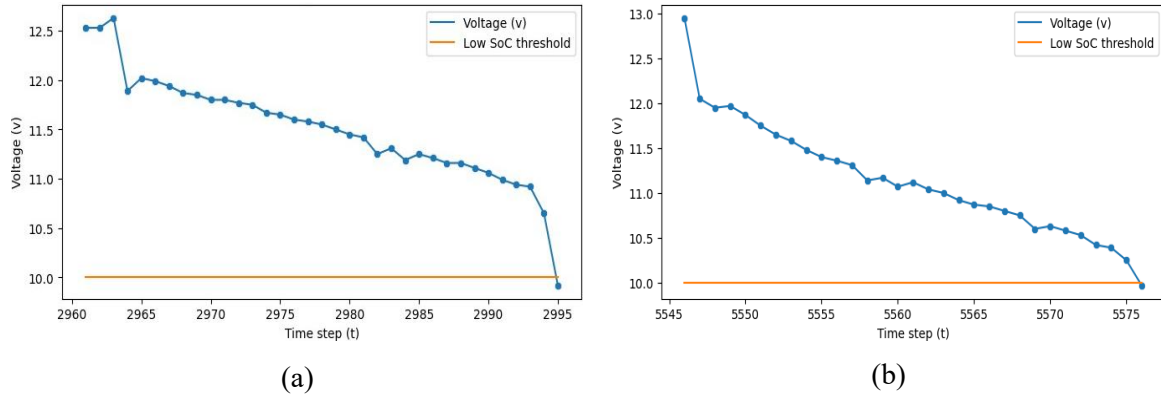


Figure 9. Battery discharge

Figure 9(a) and figure 9(b) are extracted from figure 8, specifically time-steps 2960 to 2995 and 5545 to 5575, respectively is align with figure 10. In these time-steps, the discharge voltage reaches or exceeds the low SoC threshold of 10 volts, at which point the battery connection to the electrical load is disconnected. The electrical load is transferred to another battery. This is to avoid the battery being in a deep discharge condition, which can accelerate performance degradation or even damage the battery. With VCM, real-time data in the cloud was used to detect the SoC on an active (discharging) battery. In this experiment, we used a separate microcontroller to transfer the electrical load to another battery. The round-robin algorithm was employed in this experiment; however, it is not discussed in this article.

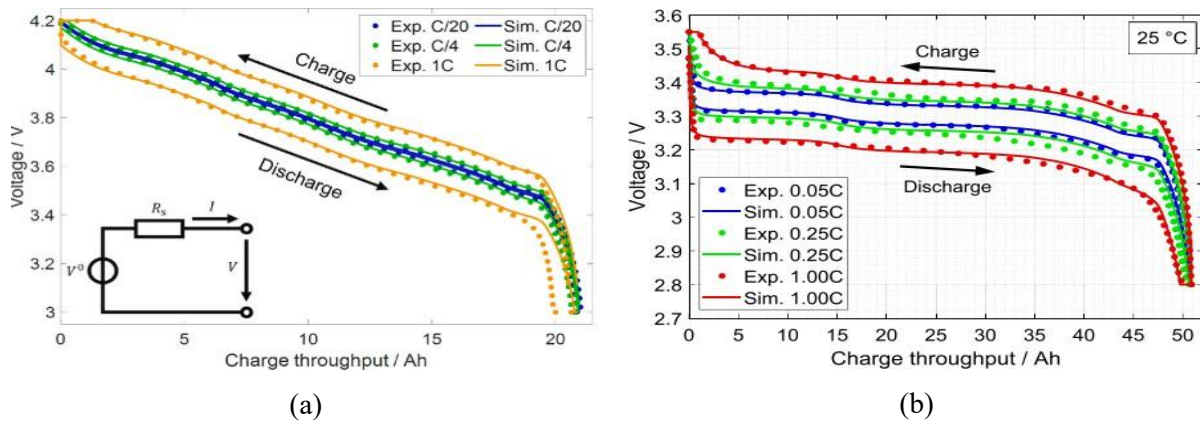


Figure 10. Battery discharge curve (a) and (b)

Sharp voltage drops over short time intervals as shown in figure 9, indicating that the battery power is in a low state, consistent with the findings presented in [32], [33], as shown in figure 10.

4.3. Data Processing for SoC Estimation

Based on the experiments, a voltage drop pattern was observed when the battery was in a discharge state. This voltage drop pattern is shown in figure 9 and is identical to the one presented in figure 10. To identify an accurate and efficient machine learning model, this work utilizes a battery discharge dataset, as shown in figure 8. The dataset is divided into 80% training data and 20% testing data, as illustrated in figure 11. ML models, including SVR, ANN, Random Forest, LSTM, and XGBoost, are used to estimate the battery's SoC. This process is carried out in Google Colabs environment using Scikit-learn, XGBoost, and TensorFlow libraries.

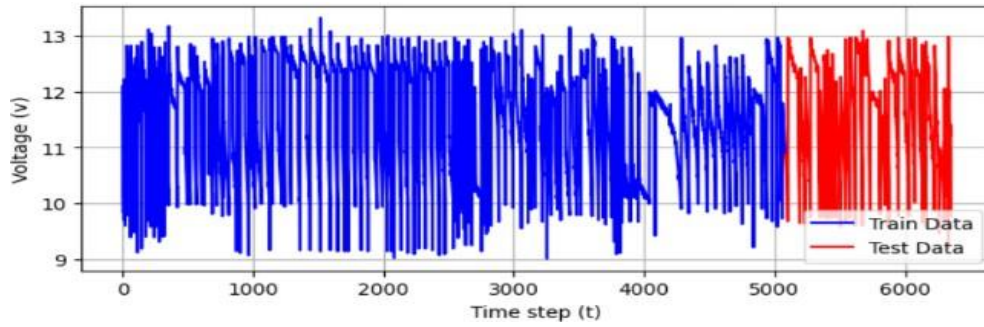


Figure 11. Train data and test data battery discharged.

Based on the pattern shown in [figure 9](#), this work uses the dataset illustrated in [figure 8](#). The estimation process is carried out using the sliding window method, which reads the previous 36 time-steps, as shown in [figure 12](#). The 36 time-steps is related to discharge pattern as shown in [figure 9](#). Comparison process with 42, 30, and 24 time-step has been done with result of higher error value. The higher value of the 42 time-step (7 hours) indicated that the value increases the complexity, while 30 time-steps (5 hours) and 24 time-steps (4 hours) did not enough to capture the discharge pattern. The pseudocode for the sliding window with a window size of 36 is presented in algorithm 1.



Figure 12. Sliding window of 36 samples

Algorithm 1. SoC prediction with a 36-window size.

Input:

- data: a sequence of values (voltage readings)
- window_size: 36 (number of past observations to use as features)

Initialize empty list $\mathbf{X}$ for features

Initialize empty list $\mathbf{y}$ for targets

For each index i from 0 to 36:

Extract a window of size 36 starting at position i

Add this window to $\mathbf{X}$

Extract the value immediately after the window (at $i + 36$)

Add this value to $\mathbf{y}$

Convert $\mathbf{X}$ and $\mathbf{y}$ to arrays

Return $\mathbf{X}$ and $\mathbf{y}$

4.4. VCM-SVR Model

The VCM-SVR model, testing was conducted with several kernels, namely rbf, poly, linear, and sigmoid, with $C=0.3$ and $\gamma = 0.01$. The test results are presented in [table 1](#), where the best performance value was achieved with the kernel set to poly, yielding MSE, MAE, and RMSE values of 0.1988, 0.2584, and 0.4617, respectively, along with a computation time of 1.113 seconds. The best performance of this model is shown in [figure 13](#).

Table 1. Error rate and computation time of VCM-SVR model

VCM-SVR model				
Kernel	MSE	MAE	RMSE	Computation time (second)
RBFB	0.2131	0.3012	0.4617	1.152
Polynomial	0.1988	0.2584	0.4459	1.113
Linear	0.2221	0.2794	4.713	31.821

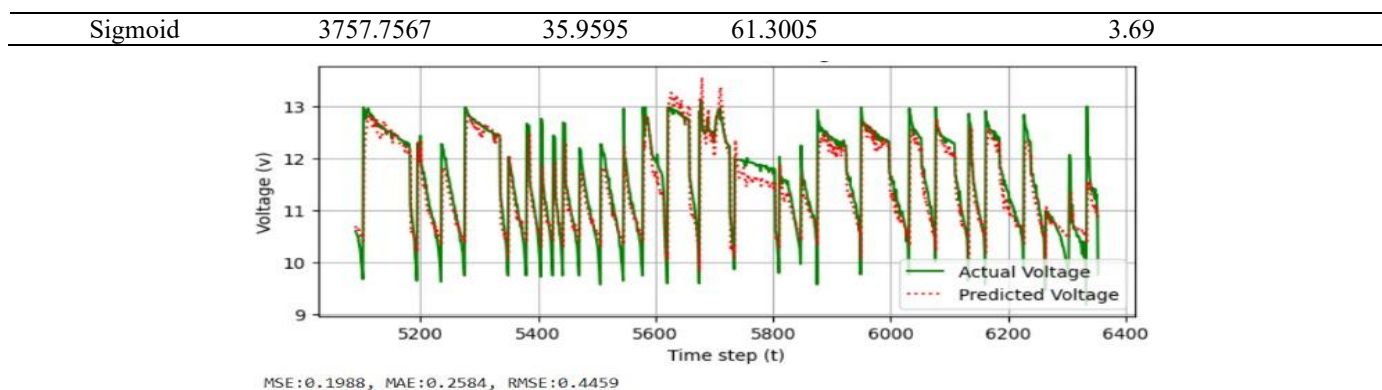


Figure 13. Actual vs Predicted SoC VCM-SVR Model

4.5. VCM-ANN Model

The VCM-ANN model testing was conducted with several epoch values: 100, 200, 300, and 400. A batch size of 16 and verbose set to 1 were used. The test results are presented in [table 2](#), where the best performance value was achieved with epoch = 100, yielding MSE, MAE, and RMSE values of 0.0886, 0.1265, and 0.2977, respectively, along with a computation time of 119.768 seconds. The best performance of this model is shown in [figure 14](#).

Table 2. Error rate and computation time of VCM-ANN model

VCM-ANN model				
Epoch	MSE	MAE	RMSE	Computation time (seconds)
100	0.0886	0.1265	0.2977	119.768
200	0.1646	0.2200	0.4057	329.67
300	0.0977	0.1450	0.3125	365.269
400	0.1040	0.1372	0.3226	497.681

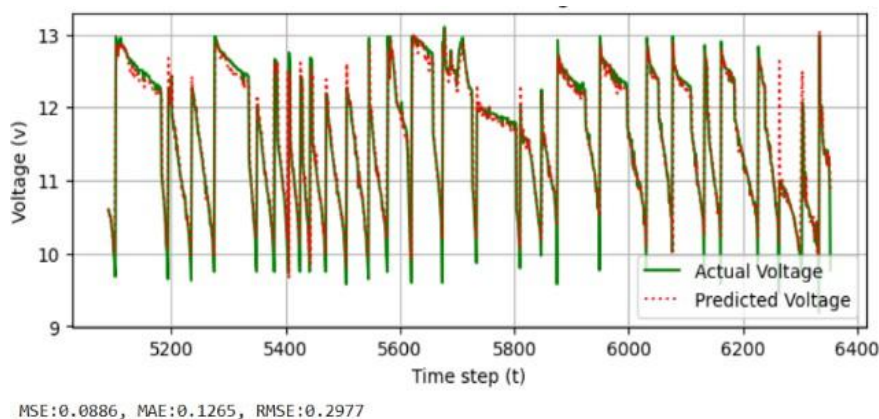


Figure 14. Actual vs Predicted SoC VCM-ANN Model

4.6. VCM-RF Model

The VCM-RF model, testing was conducted with several $n_estimator$ feature values, namely 100, 200, 300, and 400, with max_depth set to 5 and $random_state$ set to 42. The test results are presented in [table 3](#), where the best performance value was achieved with $n_estimator$ = 100, yielding MSE, MAE, and RMSE values of 0.0895, 0.1421, and 0.2992, respectively, along with a computation time of 6.129 seconds. The best performance of this model is shown in [figure 15](#).

Table 3. Error rate and computation time of VCM-RF model

VCM-RF model				
$n_estimator$	MSE	MAE	RMSE	Computation time (seconds)
100	0.0895	0.1421	0.2992	6.129

200	0.0898	0.1417	0.2996	9.116
300	0.0891	0.1419	0.2984	14.658
400	0.0898	0.1420	0.2996	20.349

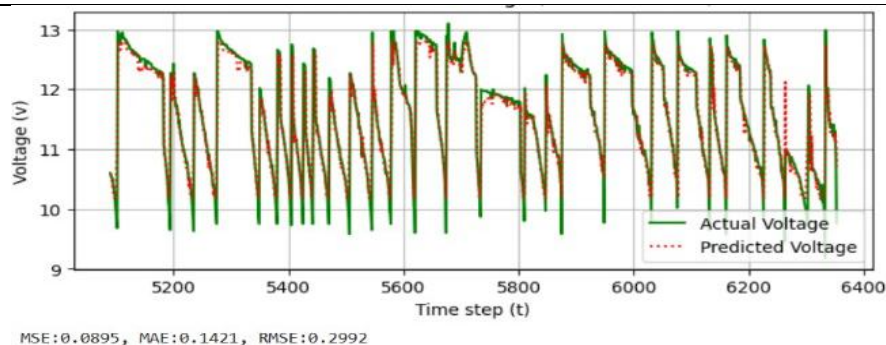


Figure 15. Actual vs Predicted SoC VCM-RF Model

4.7. VCM-LSTM Model

The VCM-LSTM model, testing was conducted with several epoch values: 50, 100, 150, and 200. The batch size was set to 16, and verbose was set to 1. The test results are presented in [table 4](#), where the best performance value was achieved with epoch = 50, yielding MSE, MAE, and RMSE values of 0.0812, 0.1223, and 0.2849, respectively, along with a computation time of 955.41 seconds. The best performance of this model is shown in [figure 16](#).

Table 4. Error rate and computation time of VCM-LSTM

VCM-LSTM model				
Epoch	MSE	MAE	RMSE	Computation time (seconds)
50	0.0812	0.1223	0.2848	955.41
100	0.0737	0.1556	0.2716	1,856.287
150	0.0897	0.1335	0.2996	2,705.403
200	0.0743	0.1384	0.2726	3,188.563

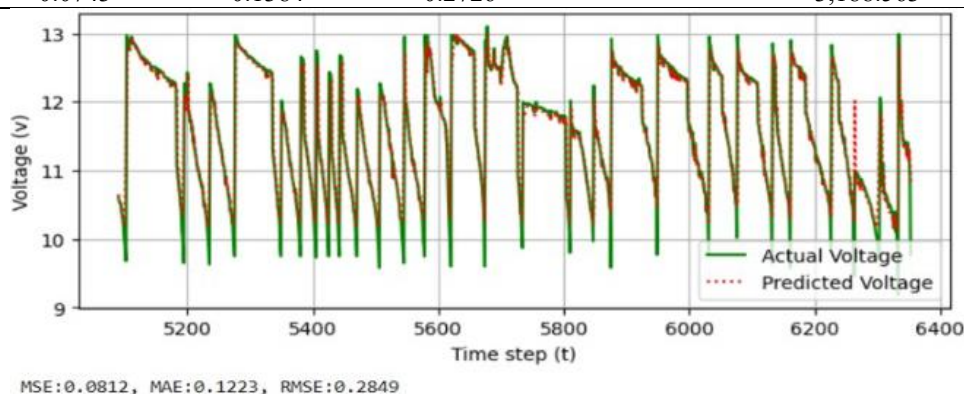


Figure 16. Actual vs Predicted SoC VCM-LSTM Model

4.8. VCM-XGBoost Model

The VCM-XGBoost model was initially tested with several $n_{estimator}$ values: 100, 200, 300, and 400, using a learning rate of 0.3 and a max_depth of 5. In this initial evaluation, the best performance was achieved with $n_{estimator} = 100$, yielding MSE, MAE, and RMSE values of 0.0849, 0.1401, and 0.2913, respectively, along with a computation time of 0.94 seconds. To obtain a more systematic and robust hyperparameter configuration, the evaluation was further extended using a Grid Search procedure. Three key hyperparameters were considered: the number of estimators, maximum tree depth, and learning rate. The number of estimators was evaluated at 100, 200, 300, and 400, the maximum tree depth at 3, 5, and 7, and the learning rate at 0.05, 0.10, 0.20, and 0.30. The optimization was performed on the training dataset using temporal cross-validation to preserve the chronological structure of the battery discharge data. The hyperparameter configuration producing the lowest validation error was selected as the optimized VCM-

XGBoost model and subsequently evaluated using the held-out test dataset [36]. The hyperparameter optimization and final evaluation results are presented in [table 5](#), while the performance of the selected model is illustrated in [figure 17](#).

Table 5. Error rate and computation time of VCM-XGBoost model

VCM-XGBoost model						
n_estimator	max_depth	Learning rate	MSE	MAE	RMSE	Computation time (seconds)
100	5	0.30	0.0849	0.1401	0.2913	0.94
200	5	0.30	0.0874	0.1448	0.2956	2.73
300	5	0.30	0.0881	0.1467	0.2968	4.23
400	5	0.30	0.0883	0.1470	0.2971	5.43

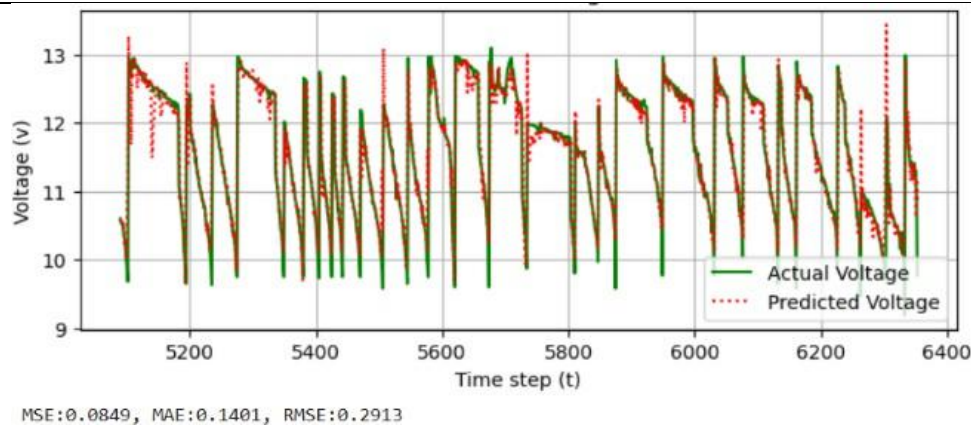


Figure 17. Actual vs Predicted SoC VCM-XGBoost Model

[Figure 17](#) illustrates the comparison between the actual and predicted SoC values obtained using the optimized VCM-XGBoost model. The predicted values generally follow the temporal pattern of the actual SoC, indicating that the optimized model is able to capture the nonlinear discharge characteristics represented by the sliding-window features.

4.9. Performance Evaluation

Based on the best-performing configuration identified for each ML model, SVR, ANN, RF, LSTM, and XGBoost were compared in terms of predictive accuracy and computational efficiency using MSE, MAE, RMSE, and computation time, as shown in [table 6](#). The comparison provides an overall assessment of the trade-off between estimation accuracy and computational cost among the evaluated VCM-ML models.

Table 6. Accuracy and efficiency comparison of the ML model

VCM-ML Model Comparison				
Ensemble model	MSE	MAE	RMSE	Computation time (seconds)
VCM-SVR	0.1988	0.2584	0.4459	1.113
VCM-ANN	0.0886	0.1265	0.2977	119.768
VCM-RF	0.0895	0.1421	0.2992	6.129
VCM-LSTM	0.0812	0.1223	0.2849	955.41
VCM-XGBoost	0.0849	0.1401	0.2913	0.94

[Table 6](#) is visualized in graphical form for easier analysis of the error and efficiency level of each proposed algorithm model. The graphs are presented in [figures 18](#).

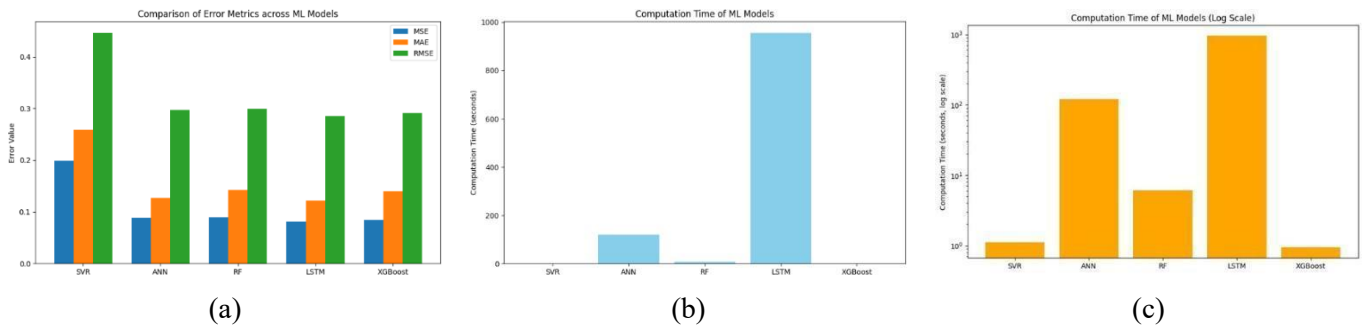


Figure 18. Comparison of prediction error metrics (a), computation time (b), and logarithmic-scale computation time (c) among the evaluated VCM-ML models.

Figure 18(a) illustrates the comparison of error metrics obtained from the evaluated VCM-ML models, all of which utilize a sliding window with 36 lag steps to represent temporal discharge patterns from the voltage signals. The results show that VCM-LSTM achieves the lowest prediction errors, with MSE = 0.0812, MAE = 0.1223, and RMSE

= 0.2849, indicating its strong capability to capture sequential dependencies within the sliding-window representation. VCM-XGBoost also demonstrates competitive predictive performance, with MSE = 0.0849, MAE = 0.1401, and RMSE = 0.2913, indicating that the structured temporal features generated by the sliding-window approach enable the tree-based model to effectively approximate nonlinear discharge behavior. In contrast, VCM-ANN and VCM-RF exhibit moderate prediction errors, while VCM-SVR produces the highest error among the evaluated models.

Figure 18(b) presents the comparison of computation time, revealing a clear trade-off between predictive accuracy and computational efficiency. Although VCM-LSTM achieves the lowest prediction errors, it requires the highest computational time of 955.41 seconds. VCM-ANN also requires substantial computation time at 119.768 seconds, whereas VCM-RF requires 6.129 seconds. In contrast, VCM-SVR and VCM-XGBoost require substantially lower computation times of 1.113 seconds and 0.94 seconds, respectively. These results indicate that VCM-XGBoost provides a favorable balance between prediction accuracy and computational efficiency among the evaluated models.

Figure 18(c), which presents the computation time in logarithmic scale, further emphasizes the substantial gap between deep learning models and lightweight ML approaches. This visualization confirms that while sliding window features enhance model input quality across all methods, the computational burden varies significantly depending on model architecture.

Overall, the integration of sliding window temporal feature extraction with VCM plays a crucial role in enabling all models to learn discharge patterns more effectively compared to raw input signals. Overall, the results indicate that VCM-XGBoost provides the most favorable trade-off between predictive accuracy and computational efficiency among the evaluated models. Its relatively low prediction error and substantially shorter computation time highlight its potential for real-time and resource-constrained BMS applications. This finding indicates that, when combined with structured time-series features, ensemble learning methods can achieve near-deep-learning accuracy while maintaining significantly lower computational cost.

5. Conclusion

This research presents a sliding window data-driven framework for SoC estimation by integrating the VCM with multiple ML algorithms. By transforming raw voltage signals into structured time-series features using a sliding window of 36 lag steps, the proposed approach effectively captures the temporal discharge dynamics of lead-acid batteries. Experimental evaluation using real-world discharge data from four batteries demonstrates that VCM-LSTM achieves the lowest prediction errors, whereas VCM-XGBoost provides the most favorable balance between predictive accuracy and computational efficiency. VCM-XGBoost achieves MSE = 0.0849, MAE = 0.1401, RMSE = 0.2913, and a computation time of 0.94 seconds, showing substantially lower computational cost while maintaining competitive prediction accuracy compared with the other evaluated VCM-ML models.

These findings indicate that the integration of sliding window feature extraction with VCM enables the evaluated ML models to represent nonlinear and time-varying battery discharge behavior using structured temporal voltage information. While deep learning approaches such as LSTM achieve slightly higher estimation accuracy, their computational cost may constrain deployment in resource-limited environments. In contrast, XGBoost provides a favorable balance between accuracy and computational efficiency, highlighting its potential as a computationally efficient model for real-time and embedded BMS applications.

Future work will extend this research toward real-world implementation and intelligent system integration. Specifically, the proposed framework will be deployed on microcontroller-based embedded platforms to evaluate real-time performance, memory usage, and energy consumption. Additionally, future studies will explore the integration of the sliding window SoC estimation model with adaptive battery management strategies, including autonomous battery switching and energy scheduling in PV-BESS systems. Further investigation will also consider robustness under varying operating conditions, such as load variability and battery aging, to ensure reliable and scalable deployment in residential and industrial applications.

6. Declarations

6.1. Author Contributions

Conceptualization: I.A.S.; Methodology: I.A.S. and E.S.; Software: I.A.S.; Validation: E.S., I.S., and I.K.T.; Formal analysis: I.A.S., E.S., I.S., and I.K.T.; Investigation: I.A.S.; Resources: I.A.S. and U.R.; Data curation: E.S., I.S., and I.K.T.; Writing original draft preparation: I.A.S., E.S., I.S., and I.K.T.; Writing, review and editing: I.A.S., E.S., I.K.T., and U.R.; Visualization: I.A.S. All authors have read and agreed to the published version of the manuscript.

6.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

6.3. Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

6.4. Institutional Review Board Statement

Not applicable.

6.5. Informed Consent Statement

Not applicable.

6.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

- [1] M. Madani, Q. Aini, I. A. Supriyono, and V. Sihotang, "Critical element for urban next-gen solar panel efficiency and adoption," *IEEE Xplore*, vol. 2024, no. Oct., pp. 1–6, 2024, doi: 10.1109/ICCIT62134.2024.
- [2] O. Tasdemir, M. Yesilbudak, and E. Irmak, "Day-ahead photovoltaic power production forecasting using a hybrid artificial neural network model integrated with metaheuristic algorithms," *Int. J. Smart Grid*, vol. 9, no. 4, pp. 1–10, 2025, doi: 10.20508/ijsmartgrid.v9i4.550.g416.
- [3] R. J. J. Molu, S. R. D. Naoussi, P. Wira, W. F. Mbasso, S. K. Tsobze, and C. Author, "Solar irradiance forecasting based on deep learning for sustainable electrical energy in Cameroon," *Int. J. Smart Grid*, vol. 7, no. 2, pp. 1–10, 2023, doi: 10.20508/ijsmartgrid.v7i2.279.g320.
- [4] S. Prakash, S. Lakshmi, S. Priya, and M. Batumalay, "Modelling and investigation of solar photovoltaic-based converter configurations with data science approach," *J. Appl. Data Sci.*, vol. 6, no. 3, pp. 1–12, 2025, doi: 10.47738/jads.v6i3.715.

- [5] R. Coteli, R. Z. Caglayan, and K. Kayisli, "Energy management based on supertwisting sliding mode in the standalone photovoltaic power system with battery backup," *Int. J. Smart Grid*, vol. 8, no. 4, pp. 1–10, 2024, doi: 10.2508/ijsmartgrid.v8i4.362.g369.
- [6] R. Sivaraman, M. H. Lin, M. I. C. Vargas, S. I. S. Al-Hawary, U. Rahardja, F. A. H. Al-Khafaji, and L. Li, "Multi-objective hybrid system development: To increase the performance of diesel/photovoltaic/wind/battery system," *Math. Model. Eng. Probl.*, vol. 11, no. 3, pp. 837–846, 2024, doi: 10.18280/mmep.110330.
- [7] A. Mohan, R. Khanna, K. Thirumala, and G. S. Ilango, "An interactive multiobjective energy management approach for grid-connected microgrids," *Comput. Electr. Eng.*, vol. 123, no. 1, pp. 1–21, 2025, doi: 10.1016/j.compeleceng.2025.110310.
- [8] R. Sarrias-Mena, P. Horrillo-Quintero, D. Carrasco-Gonz, F. Llorens-Iborra, and L. M. Fern, "Design and Raspberry Pi-based implementation of an intelligent energy management system for a hybrid AC/DC microgrid with renewable energy, battery, ultracapacitor and hydrogen system," *Comput. Electr. Eng.*, vol. 123, no. 1, pp. 1–24, 2025, doi: 10.1016/j.compeleceng.2025.110253.
- [9] S. M. Çinar, R. Doğan, and E. Akarslan, "A real-time embedded system designed for NILM studies with a novel competitive decision process algorithm," *Turk. J. Electr. Eng. Comput. Sci.*, vol. 32, no. 4, pp. 574–589, 2024, doi: 10.55730/1300-0632.4088.
- [10] A. Pati and N. Adhikary, "Optimal allocation of distributed power in adaptive droop controlled isolated DC microgrid under renewable energy intermittency and uncertainty using fuzzy logic controllers," *Comput. Electr. Eng.*, vol. 126, no. 1, pp. 1–15, 2025, doi: 10.1016/j.compeleceng.2025.110445.
- [11] J. Vadillo, A. Galarza, and F. Rodríguez, "Forecasting intra-hour solar photovoltaic energy by assembling wavelet-based time-frequency analysis with deep learning neural networks," *Int. J. Electr. Power Energy Syst.*, vol. 137, no. 1, pp. 1–12, 2022, doi: 10.1016/j.ijepes.2021.107777.
- [12] M. S. Ramadan, M. Barrera, M. Anitescu, and S. Herbert, "Simultaneous improvement of control and estimation for battery," *arXiv*, vol. 2025, no. Jul., pp. 1–10, 2025, doi: 10.48550/arXiv.2507.21300.
- [13] M. Chen, F. Han, L. Shi, Y. Feng, C. Xue, W. Gao, and Y. Xue, "Sliding mode observer for state-of-charge estimation using hysteresis-based Li-ion battery model," *Energies*, vol. 15, no. 7, pp. 1–15, 2022, doi: 10.3390/en15072658.
- [14] U. Rahardja, A. Sari, A. H. Alsalamy, S. Askar, A. H. R. Alawadi, and B. Abdullaeva, "Tribological properties assessment of metallic glasses through a genetic algorithm-optimized machine learning model," *Met. Mater. Int.*, vol. 30, no. 3, pp. 745–755, 2024, doi: 10.1007/s12540-023-01538-z.
- [15] H. Bozkurt, Ö. Çelik, and A. Teke, "Power quality enhancement in hybrid PV-BES system based on ANN-MPPT," *Turk. J. Electr. Eng. Comput. Sci.*, vol. 32, no. 5, pp. 662–681, 2024, doi: 10.55730/1300-0632.4094.
- [16] F. Syafariani, M. S. Lola, S. S. S. Abd Mutalib, W. N. F. W. Nasir, A. A. K. A. Hamid, and N. H. Zainuddin, "Leveraging a hybrid machine learning model for enhanced cyberbullying detection," *APTISI Trans. Technopreneurship*, vol. 7, no. 2, pp. 371–386, 2025, doi: 10.34306/att.v7i2.536.
- [17] W. G. Bessler, "Capacity and resistance diagnosis of batteries with voltage-controlled models," *J. Electrochem. Soc.*, vol. 171, no. 8, pp. 1–10, 2024, doi: 10.1149/1945-7111/ad6938.
- [18] R. Alberto, G. Castro, W. R. Paucar, E. Miriam, and C. Garces, "Data-driven forecasting of special education enrollment: An explainable machine learning approach," *J. Appl. Data Sci.*, vol. 7, no. 1, pp. 232–248, 2026, doi: 10.47738/jads.v7i1.1046.
- [19] L. Sitoay, M. V. A. Sin, S. Riyadi, M. Daeli, and J. Parker, "Smart urban mental health mapping through IoT sensor networks and AI analysis," *J. Orange Technol.*, vol. 1, no. 1, pp. 19–28, 2024.
- [20] V. Shelake, S. Fernandes, and S. Shrugare, "AI-driven personalized movie recommendations: A content and sentiment-aware model for streaming and digital entrepreneurship," *APTISI Trans. Technopreneurship*, vol. 7, no. 2, pp. 306–317, 2025, doi: 10.34306/att.v7i2.550.
- [21] J. Chen, Y. Zhang, J. Wu, W. Cheng, and Q. Zhu, "SOC estimation for lithium-ion battery using the LSTM-RNN with extended input and constrained output," *Energy*, vol. 262, no. 1, pp. 1–12, 2023, doi: 10.1016/j.energy.2022.125375.
- [22] M. Bendjima, A. Hazzab, M. Bechar, M. Khessam, M. Rezkallah, A. Chandra, and H. Ibrahim, "Real-time implementation of SVPWM-sensorless vector control of induction motor using an extended Kalman filter," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 29, no. 3, pp. 1402–1411, 2023, doi: 10.11591/ijeecs.v29.i3.pp1402-1411.

-
- [23] J. Chen, Y. Zhang, J. Wu, W. Cheng, and Q. Zhu, "SOC estimation for lithium-ion battery using the LSTM-RNN with extended input and constrained output," *Energy*, vol. 262, no. 1, pp. 1–12, 2023, doi: 10.1016/j.energy.2022.125375.
- [24] B. Xia, M. Ye, Q. Wang, G. Lian, Y. Li, B. Zhang, and Y. Zhang, "Robustness estimation for state-of-charge of a lithium-ion battery based on feature fusion," *Energy Rep.*, vol. 12, no. 1, pp. 4016–4025, 2024, doi: 10.1016/j.egyr.2024.09.072.
- [25] K. Yang, Y. Tang, S. Zhang, and Z. Zhang, "A deep learning approach to state of charge estimation of lithium-ion batteries based on dual-stage attention mechanism," *Energy*, vol. 244, no. 1, pp. 1–12, 2022, doi: 10.1016/j.energy.2022.123233.
- [26] M. Hardini, M. H. R. Chakim, L. Magdalena, H. Kenta, A. S. Rafika, and D. Julianingsih, "Image-based air quality prediction using convolutional neural networks and machine learning," *APTISI Trans. Technopreneurship*, vol. 5, no. 1Sp, pp. 109–123, 2023, doi: 10.34306/att.v5i1Sp.337.
- [27] N. Azizah, P. A. Sunarya, U. Rahardja, A. B. Mutiara, P. Prihandoko, and C. Pasha, "Improving smear-negative tuberculosis detection using data augmentation and Faster R-CNN," *Int. J. Cyber IT Service Manage.*, vol. 6, no. 1, pp. 65–77, 2026, doi: 10.34306/ijcitsm.v6i1.233.
- [28] X. Fan, W. Zhang, C. Zhang, A. Chen, and F. An, "SOC estimation of Li-ion battery using convolutional neural network with U-Net architecture," *Energy*, vol. 256, no. 1, pp. 1–12, 2022, doi: 10.1016/j.energy.2022.124612.
- [29] B. Rawat and R. Bhandari, "Cloud computing applications in business development," *Startupreneur Bus. Digit.*, vol. 2, no. 2, pp. 143–154, 2023, doi: 10.33050/sabda.v2i2.285.
- [30] Q. Aini, U. Rahardja, D. Manongga, I. Sembiring, M. Hardini, and H. Agustian, "IoT-based indoor air quality using ESP32," *IEEE Int. Conf. Creative Commun. Innovative Technol.*, vol. 2022, no. Nov., pp. 1–5, Nov. 2022, doi: 10.1109/ICCIT55355.2022.10119001.
- [31] T. Limouni, R. Yaagoubi, K. Bouziane, and K. Guissi, "A comprehensive review of microgrid control methods: Focus on AI, optimization, and predictive techniques," *Comput. Electr. Eng.*, vol. 125, no. 1, pp. 1–15, 2025, doi: 10.1016/j.compeleceng.2025.110442.
- [32] J. A. Braun, R. Behmann, D. Schmider, and W. G. Bessler, "State of charge and state of health diagnosis of batteries with voltage-controlled models," *J. Power Sources*, vol. 544, no. 1, pp. 1–12, Oct. 2022, doi: 10.1016/j.jpowsour.2022.231828.
- [33] J. A. Braun, R. Behmann, D. Chabrol, F. Fuchs, and W. G. Bessler, "Single-cell operando SOC and SOH diagnosis in a 24 V lithium iron phosphate battery with a voltage-controlled model," *J. Energy Storage*, vol. 85, no. 1, pp. 1–12, 2024, doi: 10.1016/j.est.2024.110986.
- [34] Wella, R. I. Desanti, and Suryasari, "A comparative study of machine learning approaches to megathrust earthquake prediction in subduction zones," *J. Appl. Data Sci.*, vol. 6, no. 4, pp. 2421–2435, 2025, doi: 10.47738/jads.v6i4.904.
- [35] A. F. Abidin, M. A. T. Mat Yusoh, A. H. Kassim, and N. H. Nik Ali, "Harmonic classification of different lighting technologies using empirical mode decomposition and support vector machines," *Turk. J. Electr. Eng. Comput. Sci.*, vol. 33, no. 3, pp. 337–356, 2025, doi: 10.55730/1300-0632.4130.
- [36] C. Slamet, R. M. Imron, A. Wahana, D. Sa, and W. B. Zulfikar, "Feature engineering for tropical rainfall forecasting using random forest and support vector regression," *J. Appl. Data Sci.*, vol. 7, no. 1, pp. 715–728, 2026, doi: 10.47738/jads.v7i1.1111.
- [37] F. Gunawan and D. A. Kristiyanti, "Optimized hybrid CNN-LSTM model for predicting transportation sector stock prices using optimizer and activation function tuning," *J. Appl. Data Sci.*, vol. 7, no. 2, pp. 1558–1571, 2026, doi: 10.47738/jads.v7i2.1323.
- [38] H. Jia, H. Sun, and S. Bai, "Decomposition LSTM with dual multihead self-attention for wind turbine drivetrain state forecasting," *Turk. J. Electr. Eng. Comput. Sci.*, vol. 33, no. 2, pp. 127–144, 2025, doi: 10.55730/1300-0632.4118.
- [39] I. G. Susrama, M. Diyasa, D. S. Alhamda, and R. L. Aryananda, "Optimizing the XGBoost model with grid search hyperparameter tuning for maximum temperature forecasting," *J. Appl. Data Sci.*, vol. 6, no. 4, pp. 2517–2529, 2025, doi: 10.47738/jads.v6i4.885.
- [40] B. Xia, M. Ye, Q. Wang, Q. Lian, Y. Li, B. Zhang, and Y. Zhang, "Robustness estimation for state-of-charge of a lithium-ion battery based on feature fusion," *Energy Rep.*, vol. 12, no. 1, pp. 4016–4025, 2024, doi: 10.1016/j.egyr.2024.09.072.
- [41] L. Semmelmann, S. Henni, and C. Weinhardt, "Load forecasting for energy communities: A novel LSTM-XGBoost hybrid model based on smart meter data," *Energy Informatics*, vol. 5, no. 1, pp. 1–21, 2022, doi: 10.1186/s42162-022-00212-9.

- [42] M. Hatta, W. N. Wahid, F. Yusuf, F. Hidayat, N. A. Santoso, and Q. Aini, "Enhancing predictive models in system development using machine learning algorithms," *Int. J. Cyber IT Service Manage.*, vol. 4, no. 2, pp. 80–87, 2024, doi: 10.34306/ijcitsm.v4i2.159.
- [43] M. C. Saputra and E. Andajani, "Analysis of factors influencing intention to adopt battery electric vehicle in Indonesia," *ADI J. Recent Innov.*, vol. 5, no. 2, pp. 100–109, 2024, doi: 10.34306/ajri.v5i2.993.
- [44] A. Mohan, R. Khanna, K. Thirumala, and G. S. Ilango, "An interactive multiobjective energy management approach for grid-connected microgrids," *Comput. Electr. Eng.*, vol. 123, no. 1, pp. 1–21, 2025, doi: 10.1016/j.compeleceng.2025.110310.
- [45] F. Cao, M. Yu, B. Liu, Z. Wei, L. Zue, M. Han, and H. Shi, "Progress of combined wind and wave energy harvesting devices and related coupling simulation techniques," *J. Mar. Sci. Eng.*, vol. 11, no. 1, pp. 1–15, Jan. 2023, doi: 10.3390/jmse11010212.
- [46] M. C. I. Geraldina, A. Muktiyanto, and U. Rahardja, "Boosting ESG performance: Overcoming collusion among entrepreneurial family and institutional shareholders," *APTISI Trans. Technopreneurship*, vol. 7, no. 1, pp. 48–60, 2025, doi: 10.34306/att.v7i1.452.
- [47] H. Tasmant, B. Bossoufi, C. Alaoui, and P. Siano, "A review of machine learning and IoT-based energy management systems for AC microgrids," *Comput. Electr. Eng.*, vol. 127, no. 1, pp. 1–15, 2025, doi: 10.1016/j.compeleceng.2025.110563.