

Electric Vehicle Trajectory Tracking Prediction Using an IPSO-Optimized Bi-LSTM Model

Mustafa Wassef Hasan¹

¹ College of Electrical Engineering, University of Technology-Iraq, Al-Sina'a Street, Baghdad 10066, Iraq.

Abstract

This study investigates electric vehicle (EV) trajectory tracking prediction using a bidirectional long short-term memory (Bi-LSTM) model optimized by an improved particle swarm optimization (IPSO) algorithm. The prediction task is formulated using model-generated driving-cycle data obtained from an EV dynamic model coupled with a brushless DC (BLDC) motor model. In this work, trajectory tracking prediction refers to the time-series response of the EV model under standardized driving-cycle profiles rather than real-time traffic-aware route planning. The considered driving-cycle profiles include nine standardized driving cycles, which are detailed in the experimental section. A one-step-ahead forecasting setting is adopted for the prediction task. The IPSO algorithm is employed to tune the Bi-LSTM hyperparameters and reduce the dependence on manual trial-and-error selection. The proposed IPSO-Bi-LSTM model is compared with long short-term memory (LSTM), gated recurrent unit (GRU), conventional Bi-LSTM, and other optimizer-linked Bi-LSTM models using mean absolute error (MAE), mean squared error (MSE), root mean square error (RMSE), and mean absolute percentage error (MAPE). The results, averaged over 10 independent runs, show that the IPSO-Bi-LSTM model achieves the lowest prediction errors among the compared models, with an MAE of 0.214 ± 0.009 , MSE of 0.102 ± 0.005 , RMSE of 0.332 ± 0.010 , and MAPE of $0.082 \pm 0.005\%$. The battery-related variables, including state of charge, current, and voltage, are reported as simulation-derived indicators associated with the predicted EV response. The present study is limited to model-generated data, and future work will focus on validation using real-world EV trajectory measurements.

Keywords: Electric vehicle, Trajectory prediction, IPSO-Bi-LSTM, Driving cycles, BLDC motor, State of charge

1. Introduction

Electric vehicles (EVs) have recently received significant attention due to advances in control, electrified power-trains, battery management, and intelligent transportation technologies. Accurate prediction of EV driving-cycle responses can support the analysis of energy consumption, state of charge (SOC), current demand, and voltage behavior under different operating profiles. In this study, the term trajectory tracking prediction refers to predicting the time-series response generated by an EV-BLDC simulation model under standardized driving cycles. It does not

refer to real-time route planning, traffic-aware navigation, or field-based trajectory forecasting.

Existing EV-related studies can be broadly grouped into path tracking control, drive-cycle prediction, battery-state estimation, and traffic-aware trajectory prediction. Path tracking control methods, such as proportional-integral-derivative (PID) [1], fractional-order PID (FOPID) [2], model predictive control (MPC) [3], linear quadratic regulator (LQR) [4], sliding mode controller [5], adaptive fuzzy PID [6], auto-tuning nonlinear PID [7], an adaptive neuro-fuzzy (ANF) with a nonlinear PID (NLPID) [8], aim to improve the dynamic tracking behavior of the

Corresponding author: Mustafa Wassef Hasan (mustafa.w.hasan@uotechnology.edu.iq)

Received: 25 April 2026; Revised: 12 September 2026; Accepted: 15 September 2026; Published: 24 September 2026

© 2026 The Author(s). This work is licensed under a Creative Commons Attribution 4.0 International License

vehicle. In contrast, sequence-learning models such as long short-term memory (LSTM) [9], convolutional neural networks (CNN) [10], gated recurrent unit (GRU) [10], and bidirectional long short-term memory (Bi-LSTM) [11] are mainly used for time-series prediction tasks. Therefore, the present work focuses on drive-cycle-based prediction using recurrent neural networks rather than on closed-loop path tracking control or traffic-aware navigation.

Time-series prediction methods for engineering applications range from classical statistical models to modern deep networks. Autoregressive integrated moving average (ARIMA) models are effective when the underlying series is adequately stationary, but their performance can deteriorate for strongly nonlinear or long-horizon dynamics [12], [13]. Recurrent neural networks (RNNs) are designed for sequential data because recurrent connections propagate temporal information across successive samples [14]. Their practical use, however, can be limited by vanishing gradients when long temporal dependencies must be learned [15]. LSTM and GRU architectures address this limitation through gated state updates [16], [17]. LSTM retains separate cell and hidden states, whereas GRU uses a more compact gating structure and generally requires fewer trainable operations. Bi-LSTM extends LSTM by processing each available input window in both temporal directions, allowing the representation to use contextual information from the beginning and end of that historical window [18], [19], [20]. In the present study, this bidirectional processing is confined to the already observed input window and does not use samples beyond the prediction origin.

A persistent practical issue across LSTM, GRU, and Bi-LSTM models is hyperparameter selection. Learning rate, hidden-layer depth, number of neurons, batch size, dropout, momentum, and related training settings can materially affect convergence and generalization [21]. Manual trial-and-error tuning is costly and can produce inconsistent configurations. Consequently, several optimization strategies have been used for recurrent-network tuning, including Bayesian optimization [21], grey wolf optimization [22], improved moth-flame optimization [23], random search [24], and improved Dung Beetle

Optimization [25]. These studies motivate the use of a systematic optimizer rather than ad hoc parameter selection, but they do not resolve how an IPSO-tuned Bi-LSTM behaves for EV responses generated by a coupled EV-BLDC model under multiple standardized driving cycles.

Recurrent and deep-learning models have been applied to several EV-related prediction tasks, including SOC estimation [26], [27], energy-consumption prediction [28], [29], charging-behavior forecasting [30], electrical-load prediction [31], battery-degradation prediction [32], and state-of-health estimation [33]. Fewer studies have examined driving-cycle trajectory-response prediction. For example, [11] used a hybrid Bi-LSTM-GRU model for EV speed prediction and energy-consumption analysis, while [34] reported multivariate LSTM speed prediction on urban-road data and compared it with ARIMA, CNN, and convolutional LSTM baselines.

The drivetrain model used to generate learning signals is also relevant to the prediction problem. Brushed DC motors have been studied in EV applications because of their controllability and high low-speed torque [35], [36], [37], but brush wear, frictional losses, thermal effects, and packaging constraints can limit their suitability [38]. DC servo motors offer compact size and accurate response [39], although parameter uncertainty and disturbances complicate controller design [40]. Induction motors provide robust construction and low maintenance [41], [42], but may exhibit higher losses and lower power factor in some operating regions [43]. BLDC motors are therefore widely investigated for EV applications because they combine high efficiency with reduced mechanical maintenance [44], [45], [46], [47], [48], [49], [50]. These considerations motivate the BLDC-based simulation platform adopted in this work.

Based on the above discussion, the research gap addressed in this work is the limited methodological evaluation of optimizer-tuned recurrent neural networks for EV driving-cycle trajectory prediction when the data are generated from a coupled EV-BLDC model. In particular, this study investigates how improved particle swarm

optimization (IPSO)-based hyperparameter tuning affects Bi-LSTM prediction accuracy under multiple standardized driving-cycle profiles. The main contributions are summarized in Table 1.

Table 1. Main contributions of the study.

No.	Contribution	Clarified scope
1	EV-BLDC-based driving-cycle data generation	The data are generated from a simulation model and are not claimed as real-world EV measurements.
2	IPSO-based Bi-LSTM hyperparameter tuning	IPSO is used to tune Bi-LSTM parameters and reduce manual trial-and-error selection.
3	Comparative prediction evaluation	The proposed model is compared with LSTM, GRU, conventional Bi-LSTM, and optimizer-linked Bi-LSTM variants.
4	Battery-status indicators	SOC, current, and voltage are reported as simulation-derived indicators associated with the predicted EV response.
5	Reproducible evaluation protocol	The study reports data splitting, evaluation metrics, and model settings for prediction assessment.

The remainder of the paper is organized as follows. Section 2 describes the EV-BLDC simulation model, Section 3 presents the recurrent networks and IPSO-based optimization procedure, Section 4 reports and discusses the prediction and computational results, and Section 5 concludes the study.

2. Electric vehicle model

Since this work focuses on EV driving-cycle trajectory prediction, the input data are generated using a physics-based EV-BLDC simulation model. The generated data are used as reference signals for training and evaluating the proposed IPSO-Bi-LSTM model. Therefore, the dataset should be interpreted as model-generated driving-cycle data rather than real-world EV measurement data.

$$F = \mu_{rr}mg + 0.5\rho_d AC_d v^2 + mg \sin \phi + m \frac{dv}{dt} \quad (1)$$

The longitudinal EV parameters used in the simulation are summarized in Table 2.

Table 2. EV model parameters used in the simulation.

Parameter	Description	Value	Unit
μ_{rr}	Rolling resistance coefficient	0.013	–
m	Vehicle mass	820	kg
g	Gravitational acceleration	9.81	m/s ²
ρ_d	Air density	1.2	kg/m ³
A	Frontal area	1.8	m ²
C_d	Aerodynamic drag coefficient	0.3	–
v	Vehicle velocity	Driving-cycle input	m/s
ϕ	Hill climbing angle	0°	Degree
r	Tire radius	0.23	m
G	Gear ratio	9.5	–

Equation (1) assumes longitudinal vehicle motion, a prescribed driving-cycle speed input, negligible lateral dynamics, and a flat-road baseline ($\phi = 0^\circ$). The parameter set in Table 2 is not presented as calibration to a specific commercial vehicle; it is a representative compact-EV simulation configuration. The rolling-resistance coefficient (0.013), drag coefficient (0.30), gravitational acceleration, and air-density assumptions are consistent with values commonly used in contemporary longitudinal EV models [51]. The mass, frontal area, tyre radius, and gear ratio are retained as study-specific simulation assumptions. The generated traction force is converted into wheel torque using Eq. (2); therefore, the EV model is used as a controlled data-generation platform rather than as a complete real-world vehicle validation model.

$$T = F \times \frac{r}{G} \quad (2)$$

where r and G are defined as the tyre radius and the gear ratio of the EV, respectively.

To quantify the dependence of the longitudinal model on

its principal assumed parameters, a local $\pm 10\%$ sensitivity calculation was performed directly from Eqs. (1)–(2) at a representative constant speed of 20 m/s, zero acceleration, and zero road grade. The baseline resistive force is 234.2 N. Table 3 reports the resulting relative change. This calculation is intended to expose model sensitivity to the assumed physical parameters; it is not a substitute for vehicle-specific parameter identification.

Table 3. Local sensitivity of the longitudinal EV model to $\pm 10\%$ parameter variation.

Parameter	−10% change	+10% change
Vehicle mass m	−4.47% in F	+4.47% in F
Rolling coefficient μ_{rr}	−4.47% in F	+4.47% in F
Drag coefficient C_d	−5.53% in F	+5.53% in F
Tyre radius r	−10.0% in T	+10.0% in T
Gear ratio G	+11.11% in T	−9.09% in T

Next, the following BLDC motor model is used to provide the EV with the required mechanical energy to move along a specified trajectory [8], [52]:

$$\begin{aligned} u_A &= Ri_A + (L - M) \frac{di_A}{dt} + e_A, \\ u_B &= Ri_B + (L - M) \frac{di_B}{dt} + e_B, \\ u_C &= Ri_C + (L - M) \frac{di_C}{dt} + e_C. \end{aligned} \quad (3)$$

The BLDC motor parameters used in the simulation are summarized in Table 4.

Table 4. BLDC motor parameters used in the simulation.

Parameter	Description	Value	Unit
i_A	Phase A current	Simulation output	A
i_B	Phase B current	Simulation output	A
i_C	Phase C current	Simulation output	A
R	Stator resistance	0.045	Ω
L	Self-inductance	0.00685	H
M	Mutual inductance	0.00162	H
e_A	Phase A back-EMF voltage	Simulation output	V
e_B	Phase B back-EMF voltage	Simulation output	V
e_C	Phase C back-EMF voltage	Simulation output	V
J	Rotor inertia	0.3	kg m ²
β	Viscous friction coefficient	0.00015	N m s/rad

The BLDC equations follow the standard phase-voltage and electromechanical formulation used in EV motor

simulations [52]. The numerical values of R , L , M , J , and β in Table 4 are study-specific simulation assumptions rather than experimentally identified parameters for a named production motor. This distinction is important because the present work evaluates prediction methodology on controlled model-generated signals, not parameter-identification accuracy for a physical drivetrain.

The electromechanical part of the BLDC motor is expressed as:

$$T_e - T_L = J \frac{d\omega_m}{dt} + \beta \omega_m \quad (4)$$

where T_L is the BLDC load torque, J is the rotor inertia, ω_m is the rotor angular velocity, β is the viscous-friction coefficient, and T_e is the electromagnetic torque, given by

$$T_e = \frac{e_A i_A + e_B i_B + e_C i_C}{\omega_m} \quad (5)$$

Figure 1 illustrates the overall model, which incorporates both vehicle and BLDC motor dynamics. These dynamics are interconnected via a transmission unit. In

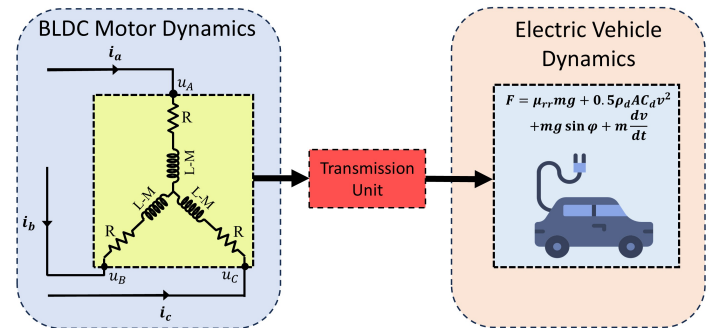


Figure 1. EV model (BLDC motor and vehicle dynamics).

addition to the predicted trajectory response, this work reports battery-related indicators, including SOC, current, and voltage. The SOC is defined as the percentage relationship between the remaining battery capacity and the nominal battery capacity under different load conditions [53]. A 4s3p lithium-ion battery pack is used in the MATLAB/Simulink environment to generate these indicators under each driving-cycle condition. Each cell has a nominal voltage of 7.2 V, a rated capacity of 5.4 Ah, and an initial SOC of 100%. These battery values

are simulation assumptions selected to provide a consistent electrical load/source model and are not manufacturer measurements. Therefore, the nominal pack capacity is $Q_n = 3 \times 5.4 = 16.2$ Ah.

The SOC is calculated using the discrete Coulomb-counting method [54]:

$$SOC_k = SOC_{k-1} - \frac{100I_{\text{dis}}(k)\Delta t}{3600Q_n}, \quad (6)$$

where SOC_k and SOC_{k-1} are the battery state of charge at the current and previous sampling instants, respectively, $I_{\text{dis}}(k)$ is the battery discharge current at sample k , Δt is the sampling time in seconds, and Q_n is the nominal battery-pack capacity in Ah. The factor 3600 converts ampere-hours into ampere-seconds, while the factor 100 expresses SOC in percentage form. In this study, the Coulombic efficiency is assumed to be unity because the SOC is used as a simulation-derived indicator rather than as an independently validated battery-aging or state-of-health estimation model.

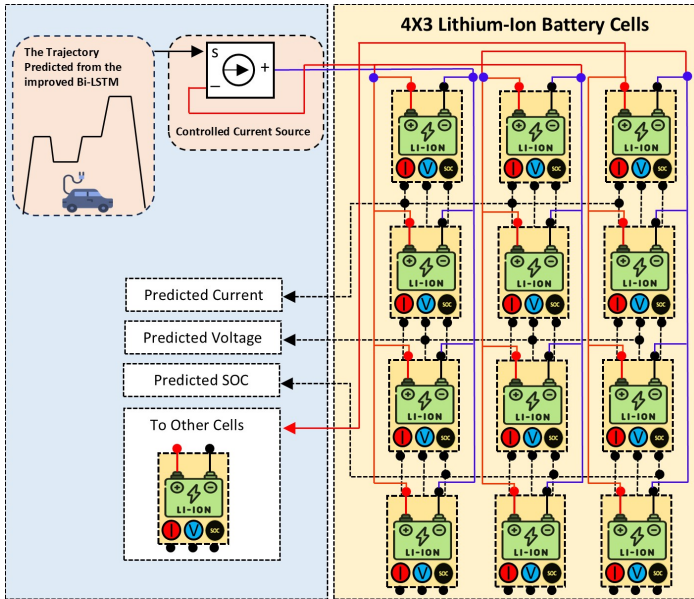


Figure 2. The simulated lithium-ion battery pack used to obtain SOC, current, and voltage indicators under different EV driving cycles.

The block-by-block signal flow is as follows. First, a standardized driving-cycle speed profile is supplied to the longitudinal vehicle block. Second, Eq. (1) converts

the commanded speed, acceleration, rolling resistance, aerodynamic drag, and road grade into traction force, and Eq. (2) maps that force to the load torque seen through the transmission. Third, the BLDC block uses this load torque with the phase-voltage and mechanical equations in Eqs. (3)–(5) to generate the simulated motor response. Fourth, the electrical demand is applied to the 4s3p battery block in Figure 2, from which pack current, voltage, and SOC are obtained; SOC is updated by Eq. (6). Finally, the simulated EV trajectory response is converted into fixed historical windows and passed to the recurrent predictor, which produces the one-step-ahead response. The corresponding battery quantities are retained as simulation-derived indicators synchronized with that response. Thus, information flows only from the prescribed driving cycle and simulated drivetrain to the predictor; no measured future traffic or future target samples are introduced.

3. Methodology

This section describes the LSTM, Bi-LSTM, and GRU prediction models, followed by the IPSO procedure used for hyperparameter optimization and the experimental protocol used for evaluation.

3.1. Bidirectional long short-term memory, long short-term memory, and gated recurrent unit

RNNs are recurrent architectures designed to propagate information across sequential observations [14]. Their standard form can suffer from vanishing gradients during back-propagation through time, which weakens the learning of long temporal dependencies [15]. LSTM was developed to mitigate this problem by introducing a gated memory cell that regulates which information is retained, updated, or released [12], [16]. This gated structure makes LSTM more suitable than a basic RNN for the driving-cycle sequences considered here.

The LSTM cell consists of an input gate i_t , a forget gate f_t , an output gate o_t , and a candidate cell state $\tilde{c}_t$. These components are controlled by sigmoid and

hyperbolic tangent activation functions. The gates regulate the information flow inside the memory cell by retaining relevant temporal information and filtering out less useful information, thereby improving sequence prediction performance [12], [55]. Each gate has two inputs: the input vector x_t at time t and the previous hidden state h_{t-1} . The subsequent equations illustrate the LSTM working principles [12], [55]:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \quad (7)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c), \quad (8)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \quad (9)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \quad (10)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \quad (11)$$

$$h_t = o_t \odot \tanh(c_t). \quad (12)$$

where x_t is the input vector at time t , h_t is the hidden state, c_t is the cell state, i_t , f_t , and o_t are the input, forget, and output gates, respectively, $\tilde{c}_t$ is the candidate cell state, W and U are the input and recurrent weight matrices, b denotes the bias vectors, $\sigma(\cdot)$ is the sigmoid activation function, $\tanh(\cdot)$ is the hyperbolic tangent function, and $\odot$ denotes element-wise multiplication.

The forget gate f_t in Eq. (9) takes values between 0 and 1 and controls how much of the previous cell state is retained. Values close to 0 suppress previous information, whereas values close to 1 preserve it. Equation (10) combines the retained cell state with the candidate update, and Eq. (11) determines the output-gate activation that regulates how the updated memory contributes to the hidden state in Eq. (12) [12], [55].

The LSTM network is a unidirectional recurrent model, where the prediction output is generated using the information available in the forward temporal direction. In contrast, the Bi-LSTM model processes each predefined input window in both forward and backward directions. Therefore, in this study, Bi-LSTM is used for offline or window-based prediction rather than instantaneous real-time route planning. The backward layer does not imply access to future traffic information during deployment;

instead, it improves contextual representation within the predefined input sequence used for each prediction sample.

The Bi-LSTM network contains a forward hidden sequence h_t^f and a backward hidden sequence h_t^b , represented by Eqs. (13) and (14), respectively. Their information is concatenated and mapped to the prediction output y_t using Eq. (15) [56].

$$h_t^f = LSTM_t^f(h_{t-1}, x_t, C_{t-1}) \quad (13)$$

$$h_t^b = LSTM_t^b(h_{t+1}, x_t, C_{t+1}) \quad (14)$$

$$y_t = \phi([h_t^f; h_t^b]) \quad (15)$$

where $\phi(\cdot)$ denotes the output mapping layer applied to the concatenated forward and backward hidden states.

Figure 3 represents the LSTM and Bi-LSTM models suggested in this work. GRU provides a more compact

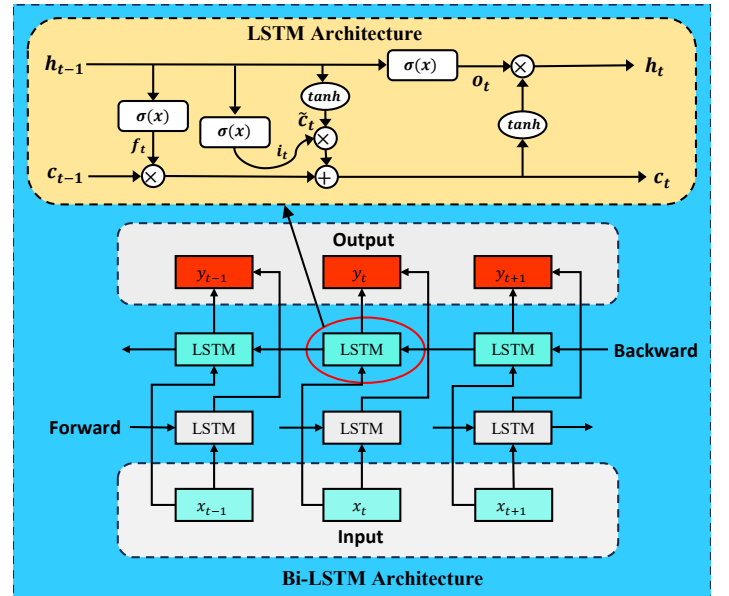


Figure 3. The Bi-LSTM and LSTM models.

recurrent architecture than LSTM by using update and reset gates and a single hidden state rather than a separate cell state. This reduced gating structure can lower the computational burden while retaining the ability to model temporal dependencies. Figure 4 shows the GRU structure, and Eqs. (16)–(19) define the update gate, reset gate, candidate hidden state, and hidden-state update, respectively [17]:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \quad (16)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), \quad (17)$$

$$\tilde{h}_t = \tanh(W_h x_t + U_h (r_t \odot h_{t-1}) + b_h), \quad (18)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t. \quad (19)$$

where z_t is the update gate, r_t is the reset gate, $\tilde{h}_t$ is the candidate hidden state, and h_t is the current hidden state.

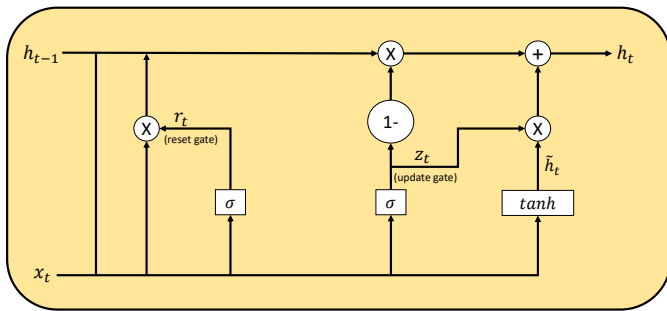


Figure 4. Schematic structure of the GRU model with update gate, reset gate, and candidate hidden state.

The LSTM, Bi-LSTM, and GRU models are evaluated using systematically selected training hyperparameters rather than manual tuning. For the LSTM/Bi-LSTM configuration, the optimized variables include hidden-layer depth, number of neurons, batch size, learning rate, dropout, momentum, and decay rate; the GRU configuration uses the applicable subset reported later in Table 7. The IPSO procedure used for this search is described in the next subsection.

3.2. Improved particle swarm optimization

Particle swarm optimization (PSO) is a population-based metaheuristic in which candidate solutions, termed particles, move through a multidimensional search space by combining inertia, individual experience, and information from the globally best solution [57]. Each particle has a position and velocity, and both are updated iteratively according to the objective-function values obtained during the search [8]. In this study, a modified PSO formulation is used to improve exploration during recurrent-network hyperparameter selection.

The PSO procedure used as the basis for IPSO is summarized in the following steps:

(1) Define the search-space dimension n and the lower and upper bounds, X_{lower} and X_{upper} . The position X_i of each of the m particles is initialized using a uniform random variable $rand() \in (0, 1)$:

$$X_i = X_{lower} + rand() \times (X_{upper} - X_{lower}) \quad (20)$$

(2) Initialize the particle velocities V_i to small values. The implementation uses the following initialization rule [8], [58]:

$$V_i = \frac{X_{lower} + rand() \times (X_{upper} - X_{lower})}{10} \quad (21)$$

(3) Evaluate the fitness value associated with each particle position.

(4) Update the personal best position ($Pbest$) of each particle whenever the current fitness improves on its previous best. The global best position ($Gbest$) is then updated if any current particle achieves a better fitness than the previous global best.

(5) Update the particle velocity using:

$$V_i^{new} = \omega V_i + rand() C_1 (Pbest - X_i) + rand() C_2 (Gbest - X_i) \quad (22)$$

where ω denotes the inertia weight and is defined by [8], [59]:

$$\omega = \frac{0.5 + rand()}{2} \quad (23)$$

Here, C_1 and C_2 denote the cognitive and social coefficients, respectively.

(6) Update the particle position according to:

$$X_i^{new} = X_i + V_i^{new} \quad (24)$$

(7) Repeat the fitness, best-position, velocity, and position updates until the maximum iteration count is

reached.

A known limitation of conventional PSO is premature convergence, in which the swarm loses diversity and becomes concentrated around a local optimum before a better region of the search space is explored [8]. This behavior is relevant here because the Bi-LSTM hyperparameter space contains mixed continuous and integer variables with nonlinear interactions. To strengthen exploration during the early iterations, the random inertia expression in Eq. (23) is replaced by the iteration-dependent form:

$$\hat{\omega} = \alpha \times \exp\left(\frac{Max_{iter} - iter}{Max_{iter}}\right) + \beta \quad (25)$$

where α is the adaptive inertia scaling coefficient, Max_{iter} is the maximum number of PSO iterations, $iter$ is the current iteration number, and β is the adaptive inertia offset.

The main IPSO configuration used for hyperparameter optimization is summarized in Table 5. These settings are kept fixed for all compared optimizer-linked models to ensure a fair computational budget. The validation mean square error (MSE) is used as the fitness function during hyperparameter optimization.

Table 5. IPSO configuration used for hyperparameter optimization and benchmark-function evaluation.

Parameter	Value	Description
Population size	40	Number of particles
Maximum iterations	500	Stopping criterion
Benchmark repetitions	250, 500, 750	Repeated runs for benchmark-function evaluation
C_1	1.5	Cognitive coefficient
C_2	2	Social coefficient
α	1.2	Adaptive inertia parameter
β	0.42	Adaptive inertia offset
Fitness function	Validation MSE	Hyperparameter objective

The population size and adaptive-inertia constants were fixed before the final comparative runs. A population of 40 particles was selected as a practical compromise between

search diversity and computational cost for the mixed continuous/integer hyperparameter space. Increasing the population directly multiplies the number of network evaluations at each iteration, whereas a much smaller population reduces the number of candidate regions explored. The pair $\alpha = 1.2$ and $\beta = 0.42$ was selected to retain comparatively stronger exploration early in the run and reduce the inertia contribution as the iteration index approaches Max_{iter} . These settings were held unchanged in every reported IPSO run so that the comparative results were not influenced by model-specific retuning. They are treated as study-specific design settings rather than universally optimal constants.

Using the adaptive inertia term, Eq. (22) is written as

$$V_i^{new} = \hat{\omega}V_i + rand()C_1(Pbest - X_i) + rand()C_2(Gbest - X_i) \quad (26)$$

To further diversify the search, a clone-selection-inspired super-mutation parameter S is incorporated into PSO [60]:

$$S = \frac{Fitness_i - Fitness_{min}}{Fitness_{max} - Fitness_{min}} + 0.1 \quad (27)$$

where $Fitness_i$ is the current particle fitness, and $Fitness_{min}$ and $Fitness_{max}$ are the best and worst fitness values in the current population, respectively.

At each update, S is compared with a uniformly distributed random number $rand_{num} \in (0, 1)$. If $S \leq rand_{num}$, the adaptive inertia $\hat{\omega}$ is used; otherwise, the original inertia ω is retained. The resulting velocity update is:

$$V_i^{new} = \begin{cases} \hat{\omega}V_i + \Delta V_i, & \text{if } S \leq rand_{num} \\ \omega V_i + \Delta V_i, & \text{otherwise} \end{cases} \quad (28)$$

where:

$$\Delta V_i = rand()C_1(Pbest - X_i) + rand()C_2(Gbest - X_i)$$

The IPSO algorithm searches for the optimal hyperpa-

rameters within predefined lower and upper bounds. Each particle in the IPSO population represents a candidate set of Bi-LSTM hyperparameters. The fitness value of each particle is evaluated using the validation mean square error (MSE), and the best particle position is selected as the optimized hyperparameter set. The search ranges used in this work are summarized in Table 6. Five multimodal

Table 6. Hyperparameter search ranges used during IPSO optimization.

Hyperparameter	Search range	Type
Learning rate	$[1 \times 10^{-4}, 1 \times 10^{-3}]$	Continuous
Epochs	[100, 300]	Integer
Batch size	[16, 128]	Integer
Dropout rate	[0.10, 0.50]	Continuous
Hidden layers	[1, 5]	Integer
Number of neurons	[20, 150]	Integer
Momentum	[0.50, 0.90]	Continuous
Decay rate	[0.90, 0.99]	Continuous

benchmark functions were used to examine the optimizer's ability to approach known global optima before applying it to recurrent-network tuning. The selected functions are the six-hump Camel function with $f^* = -1.0316$, Rastrigin with $f^* = 0$, Schaffer with $f^* = -1$, Griewank with $f^* = 0$, and Ackley with $f^* = 0$. Together, these functions provide search landscapes containing multiple local minima and different levels of multimodality [8], [61], [62]. Their mathematical definitions are:

$$f_1(x, y) = \left(4 - 2.1x^2 + \frac{x^4}{3}\right)x^2 + xy + (-4 + 4y^2)y^2$$

$$\text{for } -10 < x, y < 10$$
(29)

$$f_2(x) = \sum_{i=1}^3 (x_i^2 - 10\cos(2\pi x_i) + 10)$$

$$\text{for } -4 < x_i < 4, i = 1, 2, 3$$
(30)

$$f_3(x, y) = \frac{\sin^2 \sqrt{x^2 + y^2} - 0.5}{(1 + 0.001(x^2 + y^2))^2} - 0.5$$

$$\text{for } -4 < x, y < 4$$
(31)

$$f_4(x) = \frac{1}{4000} \sum_{i=1}^5 x_i^2 - \prod_{i=1}^5 \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$
(32)

$$\text{for } -5 < x_i < 5, i = 1, 2, \dots, 5$$

$$f_5(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{d} \sum_{i=1}^d x_i^2}\right)$$

$$- \exp\left(\frac{1}{d} \sum_{i=1}^d \cos(2\pi x_i)\right) + 20 + e$$
(33)

$$\text{for } -32.768 < x_i < 32.768, i = 1, 2, \dots, d$$

Figure 5 shows the employed multimodal benchmark functions given in this work. Next, the IPSO algorithm

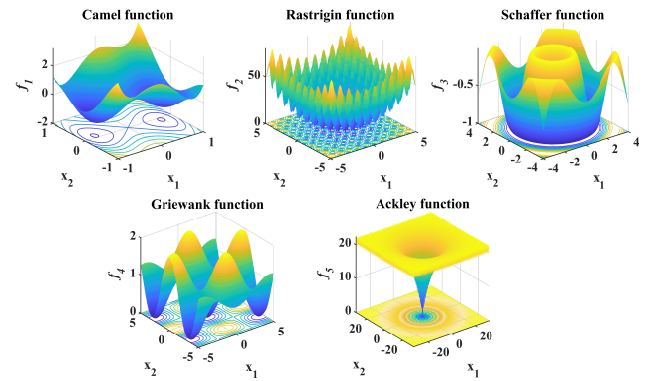


Figure 5. The Camel, Rastrigin, Schaffer, Griewank, and Ackley multimodal benchmark functions [8], [61], [62].

is evaluated using the selected multimodal benchmark functions to assess its ability to reach the global optimum or the best-known objective value. The optimizer is tested under three repeated-run settings, namely 250, 500, and 750 independent runs. For each run, the population size and maximum number of iterations are set to $n = 40$ and $t_{\max} = 500$, respectively. The obtained results are compared with IPSO based on chaotic theory (IPSO-CT) [8], conventional PSO [63], improved grey wolf optimization algorithm (IGWOA) [64], improved whale optimization algorithm based on chaotic theory (IWOA-CT) [62], genetic algorithm (GA) [65], improved sine cosine algorithm (ISCA) [61], chaos optimization

algorithm (COA) [66], and adaptive chaotic sine cosine algorithm (ACSCA) [67].

The success ratio is defined as the percentage of independent runs in which the optimizer reaches the global optimum or the best-known objective value within the predefined stopping criteria. A run is considered successful when the absolute difference between the final best objective value and the known global optimum satisfies $|f_{\text{best}} - f^*| \leq \epsilon$, where $\epsilon = 10^{-4}$ is the adopted numerical tolerance. Therefore, Figure 6 is used only to evaluate the optimizer's ability to avoid local minima on benchmark functions. The EV prediction performance is evaluated separately using the driving-cycle data and the prediction metrics reported in Section 4. Figure 6 shows the results obtained for different optimization algorithms using the Camel, Rastrigin, Schaffer, Griewank, and Ackley multimodal benchmark functions.

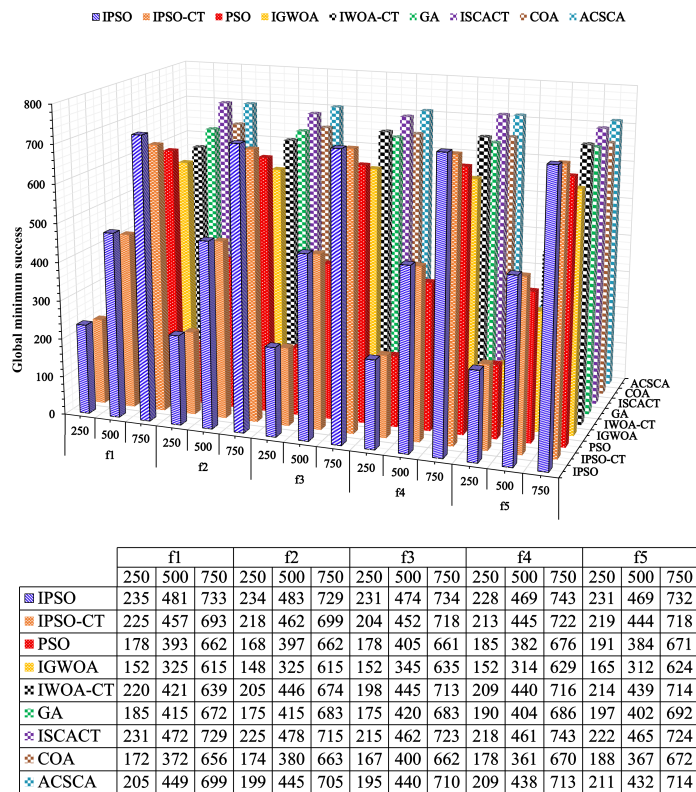


Figure 6. The obtained results for multimodal functions.

Figure 6 shows that IPSO attains the highest reported success ratios for reaching the specified global-optimum tolerance at 250, 500, and 750 repetitions. The com-

parison suggests a lower tendency toward premature convergence than the other tested optimizers under the common benchmark settings. Figure 7 summarizes how this optimizer is integrated with EV-BLDC data generation and recurrent-network prediction.

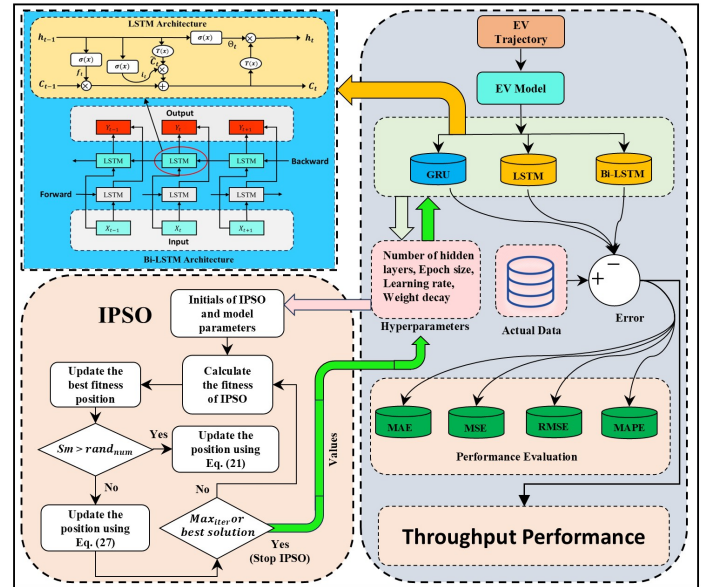


Figure 7. Overall workflow of the proposed simulation-based EV trajectory prediction framework, including EV-BLDC data generation, IPSO hyperparameter optimization, prediction-model evaluation, and execution-time assessment.

3.3. Experimental protocol

The input dataset consists of EV-BLDC simulation outputs generated under the selected standardized driving cycles. After preprocessing, the supervised learning dataset contains 16,000 input–target sequences. Each input sequence contains a fixed historical window of 20 consecutive samples, and the target is the immediately following EV trajectory-response sample; hence, the forecast horizon is one sample. Windows are formed within each driving-cycle record so that no sequence crosses from one cycle into another. The battery-related variables, including SOC, current, and voltage, are generated by the simulation model and are reported as associated indicators rather than independently validated prediction targets.

Min–max normalization is applied using parameters estimated from the training portion only, and those same scaling parameters are then applied to validation and test

data. The chronological order is preserved throughout splitting to reduce temporal leakage. The 16,000 supervised samples are divided into 11,200 training samples (70%), 2,400 validation samples (15%), and 2,400 test samples (15%). The validation set is used for IPSO fitness evaluation, whereas the test set remains untouched until final model assessment. Each model is trained and evaluated over 10 independent runs using different random initializations, and the reported results are mean $\pm$ standard deviation.

Random k -fold cross-validation was intentionally not used because it would mix temporally adjacent observations across folds and can produce optimistic estimates for sequential data. A rolling/walk-forward resampling study is also not included in the present revision; the evaluation therefore represents a fixed chronological holdout under the nine simulated driving-cycle profiles. This restriction is now stated explicitly as a limitation, and future validation will use expanding-window or rolling-origin evaluation together with measured EV trajectories.

4. Results and Discussion

After the IPSO verification, the algorithm was used to determine the optimized hyperparameters of the LSTM, Bi-LSTM, and GRU models. The optimized values are summarized in Table 7. The same training, validation, and testing protocol described in Subsection 3.3 was used for all compared models to maintain a consistent evaluation procedure. The validation mean squared error (MSE) was used as the IPSO fitness function, while the test set was used for final model assessment.

Table 7. Optimized hyperparameter values obtained using IPSO.

Hyperparameter	Bi-LSTM/LSTM	GRU
Learning rate	0.0003	0.0001
Epochs	180	210
Batch size	76	36
Dropout rate	0.20	—
Momentum	0.70	—
Decay rate	0.97	—
Hidden layers	4	—
Number of neurons	102	35

The same training, validation, and testing procedure was used for all compared models to maintain a consistent evaluation protocol. The unavailable entries in Table 7 indicate hyperparameters that were not used in the corresponding GRU configuration.

To further evaluate the optimization behavior of the proposed IPSO algorithm, the convergence profiles of the selected optimization algorithms were analyzed using the best validation MSE obtained at each iteration. The curves were averaged over 10 independent runs to reduce the effect of random initialization and provide a more stable comparison. The same Bi-LSTM structure, hyperparameter search space, population size, maximum number of iterations, and validation MSE fitness function were used for all compared optimization algorithms to ensure a fair comparison.

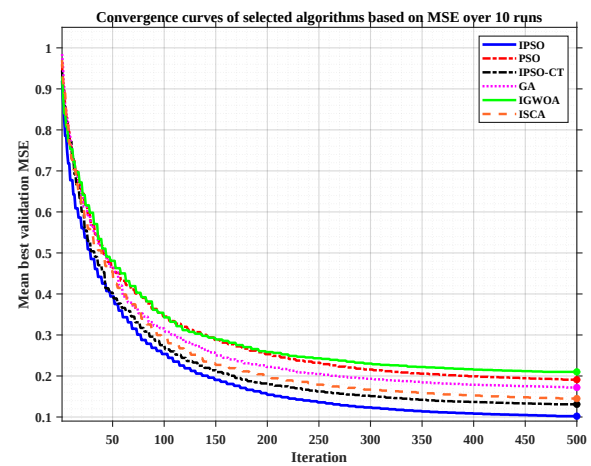


Figure 8. Mean convergence curves of the selected optimization algorithms based on the best validation MSE over 10 independent runs.

As shown in Figure 8, all algorithms gradually reduce the validation MSE as the iteration number increases. However, the proposed IPSO shows a faster reduction during the early iterations and achieves the lowest final validation MSE compared with PSO, IPSO-CT, GA, IGWOA, and ISCA. This result supports the effectiveness of the proposed IPSO in improving the hyperparameter search process for the Bi-LSTM model.

The predictive advantage should be interpreted as

the combined effect of temporal representation and hyperparameter selection rather than as an optimizer-only phenomenon. Within each 20-sample historical window, the forward LSTM summarizes the sequence in its natural order, while the backward LSTM processes the same already-observed window from its opposite end. Their concatenated representation can therefore emphasize short acceleration, deceleration, and stop–start transitions that may be weakly represented by a single directional state. IPSO simultaneously selects model capacity and training dynamics (including learning rate, batch size, dropout, depth, and neuron count), which reduces reliance on a single manually chosen configuration. The lower validation MSE in Figure 8, together with the lower test errors and small repeated-run standard deviations reported below, is consistent with this interpretation. Nevertheless, these results do not establish that bidirectionality or IPSO alone is universally superior outside the simulated driving-cycle setting.

The following equation represents the MSE [12], [68]:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (34)$$

where n is the total number of samples, y_i is the actual value, and $\hat{y}_i$ is the predicted value.

Final predictive performance on the held-out test set is evaluated using mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean square error (RMSE), defined in Eqs. (35)–(37) [12], [68].

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (35)$$

$$MAPE = \frac{100\%}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (36)$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (37)$$

Nine standardized driving-cycle profiles are used to generate the simulated EV responses: NYCC, NEDC, J-10-DDS, UDDS, J-15-DDS, HFEDS, FTP, HLA-92-DDS, and IM240 [69]. These profiles drive the EV model de-

scribed in Section 2, and the resulting trajectory-response signal forms the input/target sequence for the recurrent predictors. Synchronized SOC, current, and voltage signals are retained as battery-related simulation indicators. Figure 9 presents the NYCC results. In the displayed test segment, IPSO-Bi-LSTM tracks the simulated reference trajectory more closely than the conventional Bi-LSTM, LSTM, and GRU baselines, while the associated battery indicators remain synchronized with the predicted response.

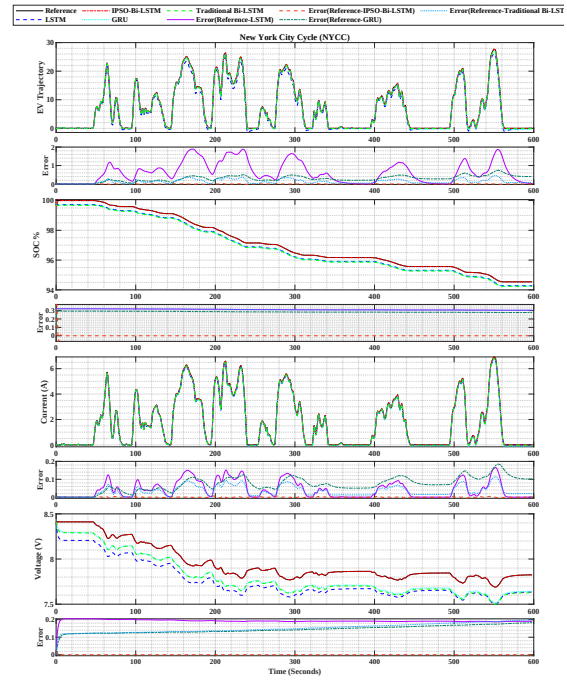


Figure 9. The predicted results for the NYCC trajectory of the IPSO-Bi-LSTM, traditional Bi-LSTM, LSTM, and GRU models.

Figures 10–17 provide the corresponding results for NEDC, J-10-DDS, UDDS, J-15-DDS, HFEDS, FTP, HLA-92-DDS, and IM240 [69]. The visual comparisons show a similar qualitative pattern across these simulated profiles. Because figure-level impressions can be sensitive to scale and segment selection, the aggregate MAE, MSE, RMSE, and MAPE values reported below are used as the primary quantitative evidence. The performance of IPSO-Bi-LSTM, traditional Bi-LSTM, LSTM, GRU, and other optimizer-linked models is assessed using MAE, MSE, RMSE, and MAPE. For each driving cycle, the metric values are calculated separately and then averaged

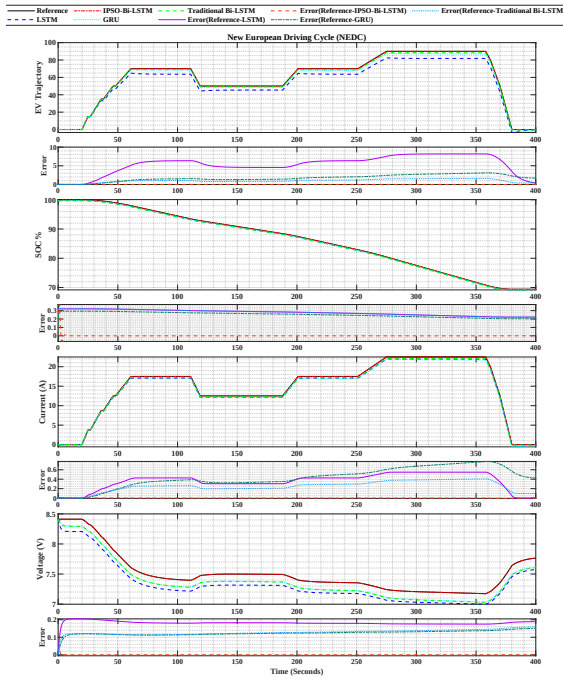


Figure 10. The predicted results for the NEDC trajectory of the IPSO-Bi-LSTM, traditional Bi-LSTM, LSTM, and GRU models.

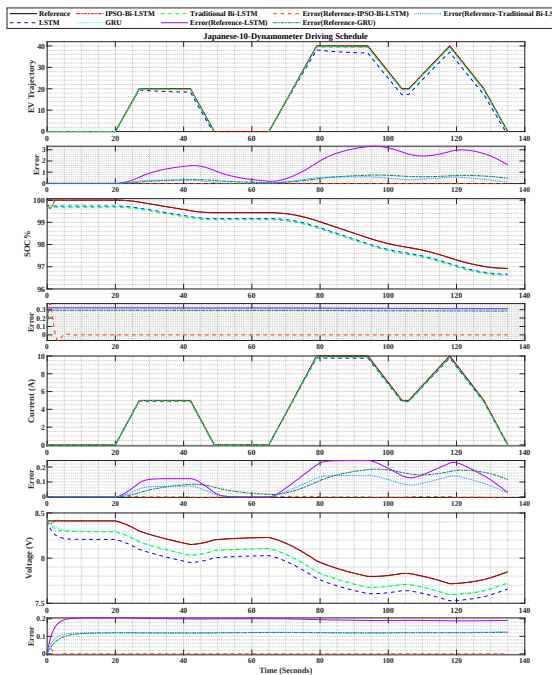


Figure 11. The predicted results for the J-10-DDS trajectory of the IPSO-Bi-LSTM, traditional Bi-LSTM, LSTM, and GRU models.

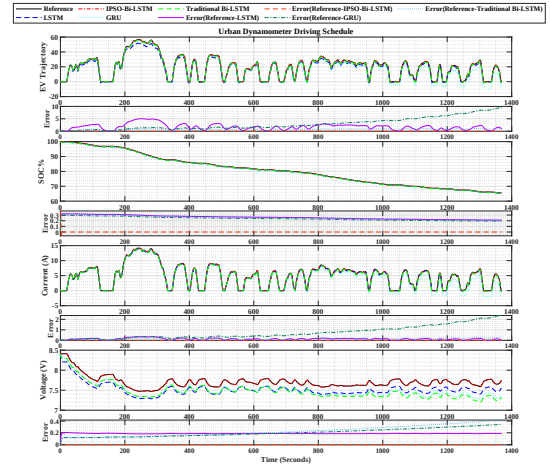


Figure 12. The predicted results for the UDDS trajectory of the IPSO-Bi-LSTM, traditional Bi-LSTM, LSTM, and GRU models.

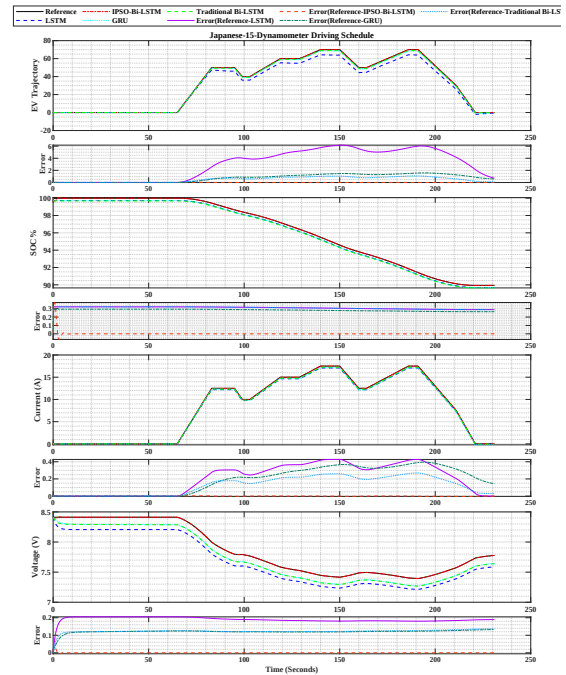


Figure 13. The predicted results for the J-15-DDS trajectory of the IPSO-Bi-LSTM, traditional Bi-LSTM, LSTM, and GRU models.

over all trajectories. Figure 18 presents the mean values of the prediction metrics over 10 independent runs, while Table 8 reports the corresponding mean $\pm$ standard deviation values.

To improve the statistical reliability of the comparison, each prediction model was trained and evaluated over 10

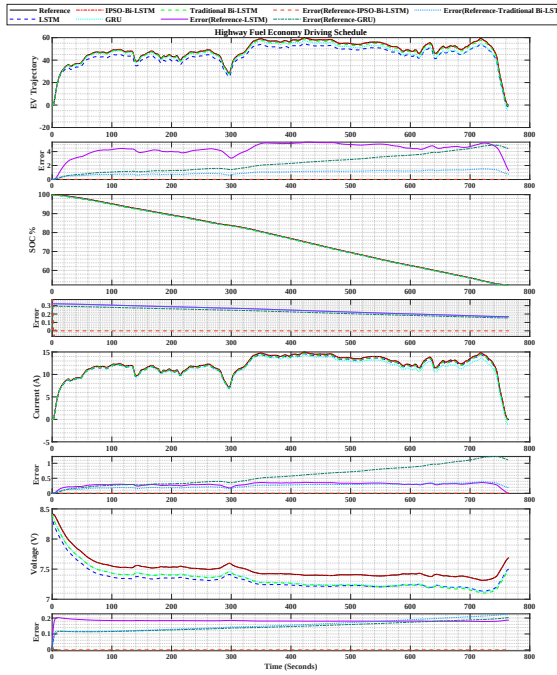


Figure 14. The predicted results for the HFEDS trajectory of the IPSO-Bi-LSTM, traditional Bi-LSTM, LSTM, and GRU models.

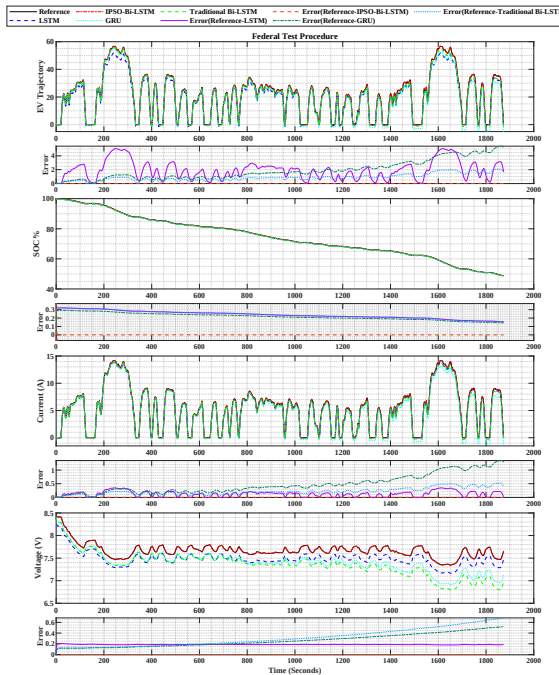


Figure 15. The predicted results for the FTP trajectory of the IPSO-Bi-LSTM, traditional Bi-LSTM, LSTM, and GRU models.

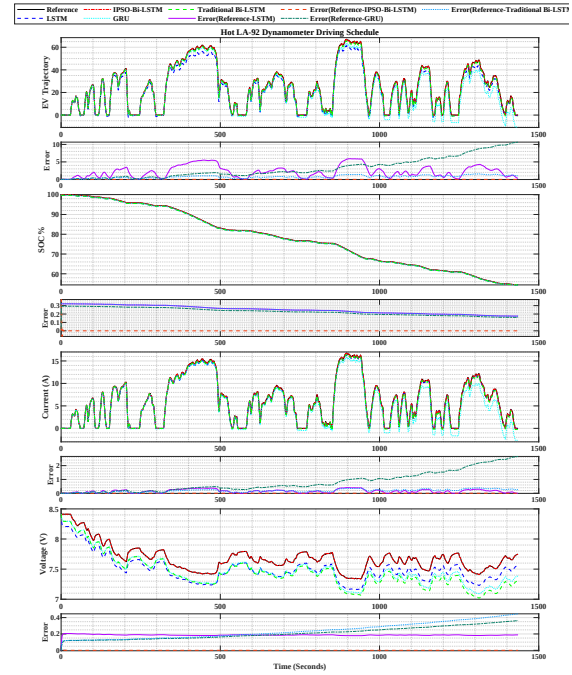


Figure 16. The predicted results for the HLA-92-DDS trajectory of the IPSO-Bi-LSTM, traditional Bi-LSTM, LSTM, and GRU models.

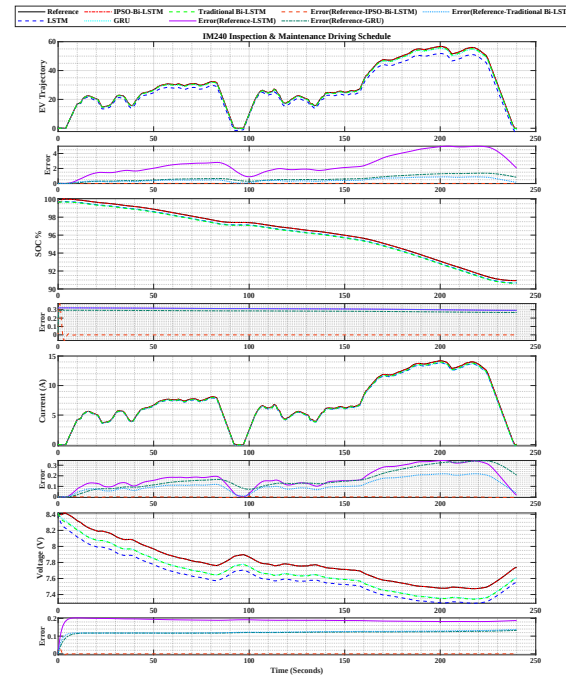


Figure 17. The predicted results for the IM240 trajectory of the IPSO-Bi-LSTM, traditional Bi-LSTM, LSTM, and GRU models.

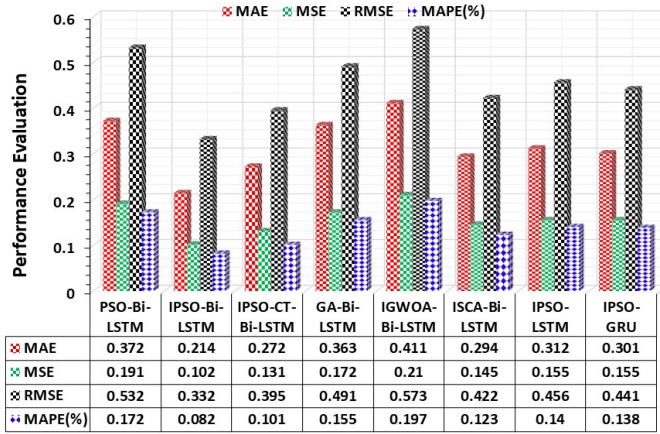


Figure 18. Mean performance evaluation of IPSO-Bi-LSTM, PSO-Bi-LSTM, IPSO-CT-Bi-LSTM, GA-Bi-LSTM, IGWOA-Bi-LSTM, ISCA-Bi-LSTM, IPSO-LSTM, and IPSO-GRU over 10 independent runs in terms of MAE, MSE, RMSE, and MAPE.

independent runs. For each run, the MAE, MSE, RMSE, and MAPE values were calculated over all considered driving cycles. Table 8 reports the mean and standard deviation values obtained from these repeated runs. For MAE, a 95% confidence interval for the mean is also reported using the Student- t interval $\bar{x} \pm t_{0.975,9}s/\sqrt{10}$.

The results in Table 8 show that the IPSO-Bi-LSTM model achieves the lowest mean values for all error metrics. In addition, the relatively small standard deviation values indicate that the proposed model maintains stable prediction performance across repeated runs. These findings support the robustness of the IPSO-based hyperparameter tuning strategy compared with the other recurrent and optimizer-linked models.

To further support the statistical comparison, a two-sided exact Wilcoxon signed-rank test was applied to paired MAE values from the 10 matched runs of IPSO-Bi-LSTM and each competing model. Runs were paired by initialization index. Zero differences, if present, are excluded by the standard signed-rank convention; no zero differences occurred in the reported comparisons, and no continuity correction or normal approximation was used. The significance level was set to $\alpha = 0.05$. In every comparison, all 10 paired MAE differences were in favor of IPSO-Bi-LSTM. Consequently, the exact two-sided probability reaches the minimum attainable value for $n =$

10, $p = 2/2^{10} = 0.001953125$, which is reported as $p = 0.0020$ after rounding. The absolute matched-pairs rank-biserial correlation is therefore $|r_{rb}| = 1.000$, indicating complete directional separation of the paired run-level MAE values.

Table 9. Wilcoxon signed-rank test results based on MAE over 10 independent runs.

Comparison	Exact p	$ r_{rb} $	Direction
IPSO-Bi-LSTM vs PSO-Bi-LSTM	0.0020	1.000	IPSO-Bi-LSTM lower
IPSO-Bi-LSTM vs IPSO-CT-Bi-LSTM	0.0020	1.000	IPSO-Bi-LSTM lower
IPSO-Bi-LSTM vs GA-Bi-LSTM	0.0020	1.000	IPSO-Bi-LSTM lower
IPSO-Bi-LSTM vs IGWOA-Bi-LSTM	0.0020	1.000	IPSO-Bi-LSTM lower
IPSO-Bi-LSTM vs ISCA-Bi-LSTM	0.0020	1.000	IPSO-Bi-LSTM lower
IPSO-Bi-LSTM vs IPSO-LSTM	0.0020	1.000	IPSO-Bi-LSTM lower
IPSO-Bi-LSTM vs IPSO-GRU	0.0020	1.000	IPSO-Bi-LSTM lower

As shown in Table 9, all paired comparisons are significant at $\alpha = 0.05$, and the effect-size direction consistently favors IPSO-Bi-LSTM. The identical rounded p -values are therefore a consequence of the same complete sign pattern across 10 paired runs rather than a reporting duplication. Statistical significance should still be interpreted together with the magnitude of the error reduction and the simulation-only scope of the data.

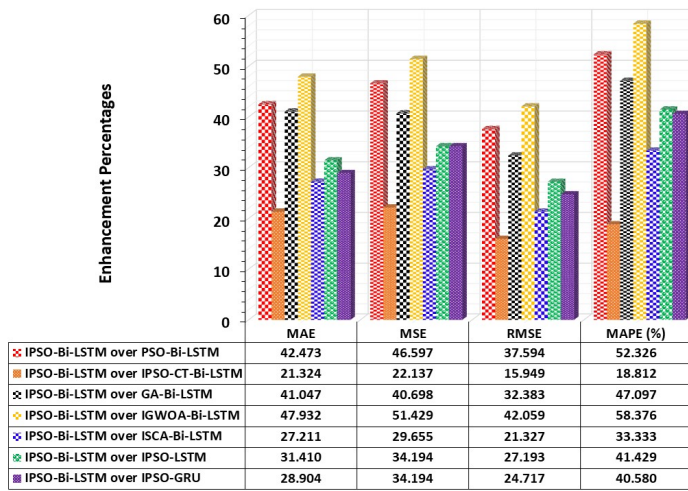
$$EP = \left(1 - \frac{M_{\text{IPSO-Bi-LSTM}}}{M_{\text{Other}}}\right) \times 100\%. \quad (38)$$

where $M_{\text{IPSO-Bi-LSTM}}$ and M_{Other} denote the mean error metric values of the proposed and compared models, respectively.

Based on the mean values reported in Table 8, the enhancement percentages of IPSO-Bi-LSTM over the compared models are calculated using Eq. (38) as shown in Figure 19. Figure 19 summarizes the relative error reductions obtained by IPSO-Bi-LSTM against each comparator. The largest gains occur against the higher-error baselines, while the smallest gains occur against the strongest optimizer-linked alternatives, consistent with the absolute metrics in Table 8.

Table 8. Average prediction metrics over all driving cycles using 10 independent runs.

Model	MAE	95% CI for mean MAE	MSE	RMSE	MAPE (%)
PSO-Bi-LSTM	0.372 ± 0.016	[0.361, 0.383]	0.191 ± 0.007	0.532 ± 0.019	0.172 ± 0.006
IPSO-Bi-LSTM	0.214 ± 0.009	[0.208, 0.220]	0.102 ± 0.005	0.332 ± 0.010	0.082 ± 0.005
IPSO-CT-Bi-LSTM	0.272 ± 0.011	[0.264, 0.280]	0.131 ± 0.009	0.395 ± 0.017	0.101 ± 0.006
GA-Bi-LSTM	0.363 ± 0.011	[0.355, 0.371]	0.172 ± 0.006	0.491 ± 0.019	0.155 ± 0.007
IGWOA-Bi-LSTM	0.411 ± 0.016	[0.400, 0.422]	0.210 ± 0.011	0.573 ± 0.020	0.197 ± 0.008
ISCA-Bi-LSTM	0.294 ± 0.011	[0.286, 0.302]	0.145 ± 0.009	0.422 ± 0.013	0.123 ± 0.008
IPSO-LSTM	0.312 ± 0.012	[0.303, 0.321]	0.155 ± 0.010	0.456 ± 0.017	0.140 ± 0.008
IPSO-GRU	0.301 ± 0.010	[0.294, 0.308]	0.155 ± 0.006	0.441 ± 0.013	0.138 ± 0.007

**Figure 19.** Enhancement percentages of the IPSO-Bi-LSTM model over the compared models based on the mean error metrics over 10 independent runs.

4.1. Computational execution time

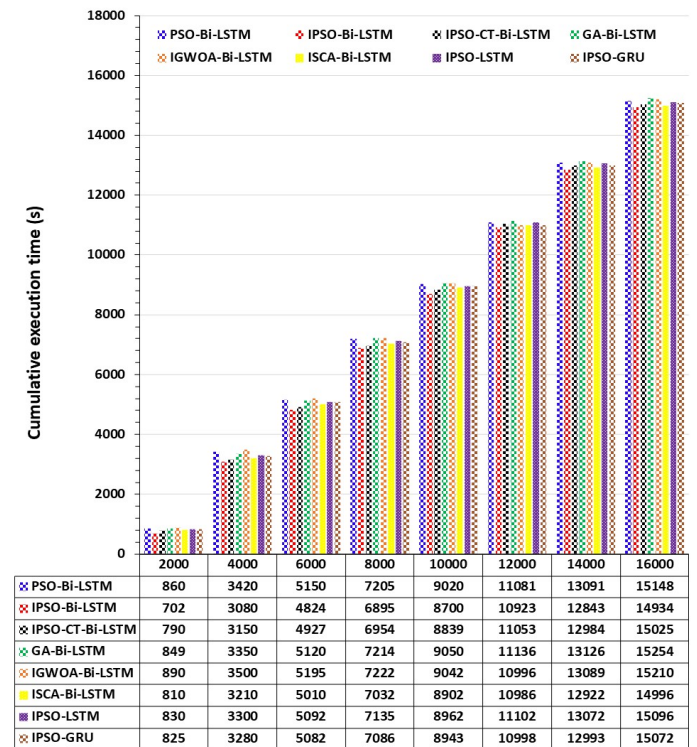
Computational execution time is also evaluated to compare the cumulative processing cost of the proposed model with the other prediction models.

$$T_{\text{exec}} = \sum_{i=1}^N t_i \quad (39)$$

where t_i is the execution time required to process the i th data interval, and N is the total number of evaluated intervals. A lower value of T_{exec} indicates lower cumulative computational cost.

All simulations were carried out using MATLAB/Simulink R2023b on a computer equipped with a 12th Gen Intel(R) Core(TM) i9-12900K CPU, 32 GB RAM, and an NVIDIA GeForce RTX 3080 GPU. The same hardware and software environment was

used for all compared models. For the execution-time experiment, the number of processed samples is increased from 2,000 to 16,000 in increments of 2,000. At each input size, the cumulative execution time is recorded for every model under the same hardware and software environment. Figure 20 reports these values and shows that IPSO-Bi-LSTM has the lowest cumulative execution time among the compared models at the evaluated input sizes. This computational result is interpreted only for the reported MATLAB/Simulink implementation and hardware configuration.

**Figure 20.** Cumulative execution time of the compared prediction models under different input data sizes.

5. Conclusion

This study presented an IPSO-optimized Bi-LSTM model for one-step-ahead EV driving-cycle trajectory prediction using signals generated by a coupled EV-BLDC simulation model. Across 10 independent runs, IPSO-Bi-LSTM achieved an MAE of 0.214 ± 0.009 , MSE of 0.102 ± 0.005 , RMSE of 0.332 ± 0.010 , and MAPE of $0.082 \pm 0.005\%$. Relative to the strongest competing optimizer-linked Bi-LSTM configuration in Table 8 (IPSO-CT-Bi-LSTM), these mean values correspond to reductions of approximately 21.3% in MAE, 22.1% in MSE, 15.9% in RMSE, and 18.8% in MAPE. The paired Wilcoxon comparisons also favored IPSO-Bi-LSTM in all 10 matched runs for every baseline ($p = 0.0020$, $|r_{rb}| = 1.000$). These results support the use of IPSO as a systematic hyperparameter-selection mechanism for Bi-LSTM within the controlled simulation setting considered here.

The practical implication is primarily methodological rather than deployment-ready: the framework can serve as a simulation-side predictor for evaluating expected EV response and synchronized battery indicators under standardized driving cycles, potentially supporting controller, energy-management, or battery-management prototyping before field testing. The reported SOC, current, and voltage signals remain simulation-derived indicators and are not independently validated battery predictions.

The main limitation is that all training and testing signals were generated by a mathematical EV-BLDC model rather than collected from instrumented vehicles. The study also uses a single fixed chronological 70/15/15 holdout and does not yet include rolling-origin validation, measured cross-vehicle transfer, or comparison with newer deep sequence architectures such as temporal convolutional networks and Transformers. Accordingly, the results should not be generalized to real-time traffic-aware route prediction or production EV deployment. Future work will combine measured EV trajectory data with expanding-window validation and will examine traffic, weather, battery aging, domain shift, and model uncertainty.

Competing Interest Statement

The author declares no competing financial, personal, or professional interests that could have influenced the work reported in this paper.

Data Availability Statement

The data used in this study were generated from the EV-BLDC simulation model described in the manuscript. The standardized driving-cycle profiles, processed simulation outputs, and implementation code used for the reported analysis are available from the corresponding author upon reasonable request.

Funding Statement

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

Author Contributions

Mustafa Wassef Hasan: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Visualization, Writing—original draft, Writing—review and editing.

Statement on the Ethical Use of AI Tools

During preparation of this manuscript, the author used ChatGPT (OpenAI; GPT-5.6 Sol) solely for language refinement. The author reviewed and edited all resulting text, independently verified the scientific content and reported results, and takes full responsibility for the content of the manuscript.

References

- [1] J. A. Morales Viscaya, A. I. Barranco Gutiérrez, and G. González Gómez, “Design, construction, and validation of an experimental electric vehicle with trajectory tracking,” *Sensors*, vol. 24, no. 9, 2024, Art. no. 2769. DOI: 10.3390/s24092769.

- [2] M. A. George, D. V. Kamat, and C. P. Kurian, "Electric vehicle speed tracking control using an ANFIS-based fractional order PID controller," *Journal of King Saud University - Engineering Sciences*, vol. 36, no. 4, pp. 256–264, 2024. DOI: 10.1016/j.jksues.2022.01.001.
- [3] L. Guo, P. Ge, M. Yue, and J. Li, "Trajectory tracking algorithm in a hierarchical strategy for electric vehicle driven by four independent in-wheel motors," *Journal of the Chinese Institute of Engineers*, vol. 43, no. 8, pp. 807–818, 2020. DOI: 10.1080/02533839.2020.1819432.
- [4] S. Wang, G. Li, J. Song, and B. Liu, "Research on an intelligent vehicle trajectory tracking method based on optimal control theory," *World Electric Vehicle Journal*, vol. 15, no. 4, 2024, Art. no. 160. DOI: 10.3390/wevj15040160.
- [5] L. A. Torres-Romero, R. Ruiz-Cruz, and L. E. González-Jiménez, "Path-following sliding mode controller for an electric vehicle considering actuator dynamics," *Machines*, vol. 12, no. 4, 2024, Art. no. 219. DOI: 10.3390/machines12040219.
- [6] X. Chen, Q. Bao, and B. Zhang, "Research on 4WIS electric vehicle path tracking control based on adaptive fuzzy PID algorithm," in *Proc. Chinese Control Conf. (CCC)*, 2019, pp. 6753–6760. DOI: 10.23919/ChiCC.2019.8865965.
- [7] M. A. Shamseldin, "Design of auto-tuning nonlinear PID tracking speed control for electric vehicle with uncertainty consideration," *World Electric Vehicle Journal*, vol. 14, no. 4, 2023, Art. no. 78. DOI: 10.3390/wevj14040078.
- [8] M. W. Hasan, A. S. Mohammed, and S. F. Noaman, "An adaptive neuro-fuzzy with nonlinear PID controller design for electric vehicles," *IFAC Journal of Systems and Control*, vol. 27, 2024, Art. no. 100238. DOI: 10.1016/j.ifacsc.2023.100238.
- [9] S. Dai, L. Li, and Z. Li, "Modeling vehicle interactions via modified LSTM models for trajectory prediction," *IEEE Access*, vol. 7, pp. 38 287–38 296, 2019. DOI: 10.1109/ACCESS.2019.2907000.
- [10] V. Singh, S. K. Sahana, and V. Bhattacharjee, "A novel CNN-GRU-LSTM based deep learning model for accurate traffic prediction," *Discover Computing*, vol. 28, 2025, Art. no. 38. DOI: 10.1007/s10791-025-09526-0.
- [11] Q. Li, R. Cheng, and H. Ge, "Short-term vehicle speed prediction based on BiLSTM-GRU model considering driver heterogeneity," *Physica A: Statistical Mechanics and its Applications*, vol. 610, 2023, Art. no. 128410. DOI: 10.1016/j.physa.2022.128410.
- [12] M. W. Hasan, "Building an IoT temperature and humidity forecasting model based on long short-term memory (LSTM) with improved whale optimization algorithm," *Memories - Materials, Devices, Circuits and Systems*, vol. 6, 2023, Art. no. 100086. DOI: 10.1016/j.memori.2023.100086.
- [13] S. Siامي-Namini, N. Tavakoli, and A. S. Namin, "The performance of LSTM and BiLSTM in forecasting time series," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2019, pp. 3285–3292. DOI: 10.1109/BigData47090.2019.9005997.
- [14] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, 2021. DOI: 10.1016/j.ijforecast.2020.06.008.
- [15] S. Srivastava and S. Lessmann, "A comparative study of LSTM neural networks in forecasting day-ahead global horizontal irradiance with satellite data," *Solar Energy*, vol. 162, pp. 232–247, 2018. DOI: 10.1016/j.solener.2018.01.005.
- [16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.
- [17] K. E. ArunKumar, D. V. Kalaga, C. M. Sai Kumar, M. Kawaji, and T. M. Brenza, "Comparative analysis of gated recurrent units (GRU), long short-term memory (LSTM) cells, autoregressive integrated moving average (ARIMA), seasonal autoregressive integrated moving average (SARIMA) for forecasting COVID-19 trends," *Alexandria Engineering Journal*, vol. 61, no. 10, pp. 7585–7603, 2022. DOI: 10.1016/j.aej.2022.01.011.
- [18] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values," *Transportation Research Part C: Emerging Technologies*, vol. 118, 2020, Art. no. 102674. DOI: 10.1016/j.trc.2020.102674.
- [19] D. Gao, X. Liu, Z. Zhu, and Q. Yang, "A hybrid CNN-BiLSTM approach for remaining useful life prediction of EVs lithium-ion battery," *Measurement and Control*, vol. 56, no. 1–2, pp. 371–383, 2023. DOI: 10.1177/00202940221103622.
- [20] J. Liu et al., "A study of electric bicycle lithium battery charging monitoring using CNN and BiLSTM

- networks model with NILM method,” *Electronics*, vol. 13, no. 16, 2024, Art. no. 3316. DOI: 10.3390/electronics13163316.
- [21] S. Vedhanayaki and V. Indragandhi, “A bayesian optimized deep learning approach for accurate state of charge estimation of lithium ion batteries used for electric vehicle application,” *IEEE Access*, vol. 12, pp. 43 308–43 327, 2024. DOI: 10.1109/ACCESS.2024.3380188.
- [22] G. Taş, C. Bal, and A. Uysal, “Performance comparison of lithium polymer battery SOC estimation using GWO-BiLSTM and cutting-edge deep learning methods,” *Electrical Engineering*, vol. 105, no. 5, pp. 3383–3397, 2023. DOI: 10.1007/s00202-023-01934-z.
- [23] Z. Wang, H. Zhang, B. Li, X. Fan, Z. Ma, and J. Zhou, “An IMFO-LSTM_BiGRU combined network for long-term multiple battery states prediction for electric vehicles,” *Energy*, vol. 309, 2024, Art. no. 133069. DOI: 10.1016/j.energy.2024.133069.
- [24] J. Siddiqui et al., “Electric vehicle charging station load forecasting with an integrated DeepBoost approach,” *Alexandria Engineering Journal*, vol. 116, pp. 331–341, 2025. DOI: 10.1016/j.aej.2024.12.069.
- [25] G. Liu and X. Zhou, “Electric vehicle load prediction based on CNN-BiLSTM and XGBoost combined model,” in *Proc. China Autom. Congr. (CAC)*, 2024, pp. 2601–2606. DOI: 10.1109/CAC63892.2024.10865472.
- [26] M. K. Khan, M. A. Houran, K. Kauhaniemi, M. H. Zafar, M. Mansoor, and S. Rashid, “Efficient state of charge estimation of lithium-ion batteries in electric vehicles using evolutionary intelligence-assisted GLA-CNN-bi-LSTM deep learning model,” *Heliyon*, vol. 10, no. 15, 2024, Art. no. e35183. DOI: 10.1016/j.heliyon.2024.e35183.
- [27] B. Devi, V. S. 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*, vol. 100, 2024, Art. no. 113622. DOI: 10.1016/j.est.2024.113622.
- [28] X. Zhang, Z. Chen, W. Wang, and X. Fang, “Prediction method of PHEV driving energy consumption based on the optimized CNN BiLSTM attention network,” *Energies*, vol. 17, no. 12, 2024, Art. no. 2959. DOI: 10.3390/en17122959.
- [29] N. Klungsida, P. Maneechot, N. Butploy, and K. Khiewwan, “Forecasting energy consumption from EV station charging using RNN, LSTM and GRU neural network,” *Journal of Renewable Energy and Smart Grid Technology*, vol. 19, no. 1, pp. 1–6, 2024. DOI: 10.69650/rast.2024.254636.
- [30] B. M. Fahim, M. K. Akbar, and M. Amayri, “ResiDual-Net: A novel electric vehicle charging data imputation technique to enhance load forecasting accuracy,” *Building Simulation*, vol. 18, no. 4, pp. 897–922, 2025. DOI: 10.1007/s12273-025-1236-8.
- [31] C. Pavlatos, E. Makris, G. Fotis, V. Vita, and V. Mladenov, “Enhancing electrical load prediction using bidirectional LSTM neural network,” *Electronics*, vol. 12, no. 22, 2023, Art. no. 4652. DOI: 10.3390/electronics12224652.
- [32] J. Guo, M. Liu, P. Luo, X. Chen, H. Yu, and X. Wei, “Attention-based BiLSTM for the degradation trend prediction of lithium battery,” *Energy Reports*, vol. 9, pp. 655–664, 2023. DOI: 10.1016/j.egy.2023.03.056.
- [33] D. Kumar, M. Rizwan, and A. K. Panwar, “Robust state of health estimation of commercial lithium-ion batteries based on enhanced hybrid machine learning model for electrified transportation,” *Electrical Engineering*, vol. 107, no. 4, pp. 5053–5070, 2025. DOI: 10.1007/s00202-024-02808-8.
- [34] Y. Naitmalek, M. Najib, M. Bakhouya, J. Gaber, and M. Essaaidi, “Embedded real-time speed forecasting for electric vehicles: A case study on RSK urban roads,” *IEEE Access*, vol. 10, pp. 126 412–126 428, 2022. DOI: 10.1109/ACCESS.2022.3225643.
- [35] D. Baidya, S. Dhopte, and M. Bhattacharjee, “Sensing system assisted novel PID controller for efficient speed control of DC motors in electric vehicles,” *IEEE Sensors Letters*, vol. 7, no. 1, pp. 1–4, 2023. DOI: 10.1109/LSENS.2023.3234400.
- [36] M.-H. Chang and Y.-C. Wu, “Speed control of electric vehicle by using type-2 fuzzy neural network,” *International Journal of Machine Learning and Cybernetics*, vol. 13, no. 6, pp. 1647–1660, 2022. DOI: 10.1007/s13042-021-01475-6.
- [37] M. M. Hussain, Z. A. Memon, M. A. Chaudhary, and M. Siddique, “An innovative PID controller in conjunction with DC electric motor for control of hybrid electric vehicle,” *SSRG Int. J. Electr. Electron. Eng.*, vol. 7, no. 7, pp. 20–34, 2020. DOI: 10.14445/23488379/IJEEE-V7I7P105.
- [38] N. Hashemnia and B. Asaei, “Comparative study of using different electric motors in the electric vehicles,” in *Proc. 18th Int. Conf. Electr. Mach. (ICEM)*, 2008, pp. 1–5. DOI: 10.1109/ICELMACH.2008.4800157.

- [39] L. Li, X. Wang, X. Qi, X. Li, D. Cao, and Z. Zhu, "Automatic clutch control based on estimation of resistance torque for AMT," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 6, pp. 2682–2693, 2016. DOI: 10.1109/TMECH.2016.2517088.
- [40] X. Wang, W. Wang, L. Li, J. Shi, and B. Xie, "Adaptive control of DC motor servo system with application to vehicle active steering," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 3, pp. 1054–1063, 2019. DOI: 10.1109/TMECH.2019.2906250.
- [41] H. Salahuddin et al., "Electric vehicle transient speed control based on vector control FM-PI speed controller for induction motor," *Applied Sciences*, vol. 12, no. 17, 2022, Art. no. 8694. DOI: 10.3390/app12178694.
- [42] M. S. Sepeeh, S. A. Zulkifli, S. Y. Sim, H.-J. Chiu, and M. Z. C. Wanik, "Speed tracking for IFOC induction motor speed control using hybrid sensorless speed estimator based on flux error for electric vehicles application," *Machines*, vol. 10, no. 11, 2022, Art. no. 1089. DOI: 10.3390/machines10111089.
- [43] M. Yildirim, M. Polat, and H. Kurum, "A survey on comparison of electric motor types and drives used for electric vehicles," in *Proc. PEMC*, 2014, pp. 218–223. DOI: 10.1109/EPEPMC.2014.6980715.
- [44] A. Intidam et al., "Development and experimental implementation of optimized PI-ANFIS controller for speed control of a brushless DC motor in fuel cell electric vehicles," *Energies*, vol. 16, no. 11, 2023, Art. no. 4395. DOI: 10.3390/en16114395.
- [45] M. E. A. Mohamed and Y. Guo, "Design of speed control for brushless DC motor used for electric vehicle based on adaptive neuro-fuzzy inference system," *IOP Conference Series: Materials Science and Engineering*, vol. 612, no. 4, 2019, Art. no. 042066. DOI: 10.1088/1757-899X/612/4/042066.
- [46] R. Ramachandran, D. Ganeshaperumal, and B. Subathra, "Closed-loop control of BLDC motor in electric vehicle applications," in *Proc. INCCES*, 2019, pp. 1–5. DOI: 10.1109/INCCES47820.2019.9167730.
- [47] M. S. Veerajju, "Optimal tuning of PID controller using bio-inspired algorithm for speed control of brushless DC motor in electric vehicle," *International Journal of Mechanical Engineering*, vol. 6, no. 3, pp. 3463–3470, 2021.
- [48] M. Ramesh, G. Rao, J. Amarnath, S. Kamakshaiah, and B. Jawaharlal, "Speed torque characteristics of brushless DC motor in either direction on load using ARM controller," in *Proc. IEEE PES Innov. Smart Grid Technol.-India (ISGT India)*, 2011, pp. 217–222. DOI: 10.1109/ISET-India.2011.6145385.
- [49] X. D. Xue, K. W. E. Cheng, and N. C. Cheung, "Selection of electric motor drives for electric vehicles," in *Proc. Australas. Univ. Power Eng. Conf. (AUPEC)*, Art. no. 4813059, 2008, pp. 1–6.
- [50] S. R. Jape and A. Thosar, "Comparison of electric motors for electric vehicle application," *Int. J. Res. Eng. Technol.*, vol. 6, no. 9, pp. 12–17, 2017. DOI: 10.15623/ijret.2017.0609004.
- [51] A. Babangida and P. T. Szemes, "Dynamic modeling and control strategy optimization of a volkswagen crafter hybrid electrified powertrain," *Energies*, vol. 17, no. 18, p. 4721, 2024. DOI: 10.3390/en17184721.
- [52] A. Jaya, E. Purwanto, M. B. Fauziah, F. D. Murdianto, G. Prabowo, and M. R. Rusli, "Design of PID-fuzzy for speed control of brushless DC motor in dynamic electric vehicle to improve steady-state performance," in *Proc. Int. Electron. Symp. Eng. Technol. Appl. (IES-ETA)*, 2017, pp. 179–184. DOI: 10.1109/ELECSYM.2017.8240399.
- [53] E. O. Ofoegbu, "State of charge (SOC) estimation in electric vehicle (EV) battery management systems using ensemble methods and neural networks," *Journal of Energy Storage*, vol. 114, 2025, Art. no. 115833. DOI: 10.1016/j.est.2025.115833.
- [54] K. Movassagh, A. Raihan, B. Balasingam, and K. Pattipati, "A critical look at coulomb counting approach for state of charge estimation in batteries," *Energies*, vol. 14, no. 14, 2021, Art. no. 4074. DOI: 10.3390/en14144074.
- [55] M. W. Hasan, "Design of an IoT model for forecasting energy consumption of residential buildings based on improved long short-term memory (LSTM)," *Measurement: Energy*, vol. 5, 2025, Art. no. 100033. DOI: 10.1016/j.meaeene.2024.100033.
- [56] M. S. Jassim, G. Coskuner, N. Sultana, and S. M. Z. Hos-sain, "Forecasting domestic waste generation during successive COVID-19 lockdowns by bidirectional LSTM super learner neural network," *Applied Soft Computing*, vol. 133, 2023, Art. no. 109908. DOI: 10.1016/j.asoc.2022.109908.
- [57] H. Qiu, W. Yu, G. Zhang, X. Xia, and K. Yao, "Multi-robot collaborative 3D path planning based on game theory and particle swarm optimization hybrid method," *The Journal of Supercomputing*, vol. 81, 2025, Art. no. 487. DOI: 10.1007/s11227-025-06960-1.
- [58] V. Steffen, "Particle swarm optimization with a simplex strategy to avoid getting stuck on local optimum," *AI*,

- Computer Science and Robotics Technology*, vol. 2022, pp. 1–40, 2022. DOI: 10.5772/acrt.11.
- [59] R. C. Eberhart and Y. Shi, “Tracking and optimizing dynamic systems with particle swarms,” in *Proc. Congr. Evol. Comput. (CEC)*, vol. 1, 2001, pp. 94–100. DOI: 10.1109/CEC.2001.934376.
- [60] L. N. de Castro and F. J. Von Zuben, “The clonal selection algorithm with engineering applications,” in *Proc. GECCO Workshop Artif. Immune Syst. Appl.*, 2000, pp. 36–37.
- [61] M. W. Hasan and L. G. Ibrahim, “A pursuit-evasion game robot controller design based on a neural network with an improved optimization algorithm,” *Results in Control and Optimization*, vol. 17, 2024, Art. no. 100503. DOI: 10.1016/j.rico.2024.100503.
- [62] M. W. Hasan, “Disturbance rejection controller design based on adaptive nonlinear FOPID controller and chaotic WOA with a neuro-fuzzy approximation for URV robot,” *E-Prime - Advances in Electrical Engineering, Electronics and Energy*, vol. 5, 2023, Art. no. 100280. DOI: 10.1016/j.prime.2023.100280.
- [63] F. Marini and B. Walczak, “Particle swarm optimization (PSO). a tutorial,” *Chemometrics and Intelligent Laboratory Systems*, vol. 149, pp. 153–165, 2015. DOI: 10.1016/j.chemolab.2015.08.020.
- [64] M. W. Hasan and N. H. Abbas, “An adaptive neural sliding mode controller design for autonomous underwater vehicle based on improved grey wolf optimization algorithm,” *Journal of Engineering Science and Technology Review*, vol. 15, no. 1, pp. 100–109, 2022. DOI: 10.25103/jestr.151.13.
- [65] Y. Mitsukura, T. Yamamoto, and M. Kaneda, “A design of self-tuning PID controllers using a genetic algorithm,” in *Proc. Amer. Control Conf. (ACC)*, vol. 2, 1999, pp. 1361–1365. DOI: 10.1109/ACC.1999.783590.
- [66] M. S. Tavazoei and M. Haeri, “An optimization algorithm based on chaotic behavior and fractal nature,” *Journal of Computational and Applied Mathematics*, vol. 206, no. 2, pp. 1070–1081, 2007. DOI: 10.1016/j.cam.2006.09.008.
- [67] Y. Ji et al., “An adaptive chaotic sine cosine algorithm for constrained and unconstrained optimization,” *Complexity*, vol. 2020, 2020, Art. no. 6084917. DOI: 10.1155/2020/6084917.
- [68] N. Somu, M. R. Gauthama Raman, and K. Ramamritham, “A hybrid model for building energy consumption forecasting using long short term memory networks,” *Applied Energy*, vol. 261, 2020, Art. no. 114131. DOI: 10.1016/j.apenergy.2019.114131.
- [69] United States Environmental Protection Agency. “Dynamometer drive schedules,” United States Environmental Protection Agency, Accessed: Aug. 2, 2026. [Online]. Available: <https://www.epa.gov/vehicle-and-fuel-emissions-testing/dynamometer-drive-schedules>