Open Access Research Journal of
Engineering and Technology
Journals home page: https://oarjpublication/journals/oarjet/
ISSN: 2783-0128 (Online)

(RESEARCH ARTICLE)

CorrACC-guided feature selection and metaheuristic hyperparameter tuning for deep learning in IoT Environmental monitoring

Heydar Fahem Khenzir and Peyman Neamatollahi *

Department of Computer Engineering, Faculty of Electrical and Computer Engineering, Hakim Sabzevari University, Sabzevar, Iran.

Open Access Research Journal of Engineering and Technology, 2026, 10(02), 037-057

Publication history: Received on 25 March 2026; revised on 24 April 2026; accepted on 27 April 2026

Article DOI: https://doi.org/10.53022/oarjet.2026.10.2.0027

Abstract

The Internet of Things (IoT) has significantly advanced environmental monitoring by enabling continuous data acquisition from distributed sensor networks. However, challenges such as high-dimensional feature spaces, redundant measurements, and limited computational resources hinder the efficiency and reliability of predictive models. This study proposes an integrated learning framework that combines correlation-aware feature selection (CorrACC), metaheuristic hyperparameter optimization, and deep learning for accurate and efficient environmental prediction. The framework systematically evaluates both classical machine learning models (KNN, Decision Tree, Random Forest, and Lasso) and deep learning architectures (CNN, RNN, LSTM, and Bi-LSTM) under a unified experimental setup. To enhance model performance, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Differential Evolution (DE) are employed for automated hyperparameter tuning. Experimental results demonstrate that deep learning models consistently outperform traditional approaches, with Bi-LSTM achieving the highest predictive accuracy across multiple evaluation metrics. Furthermore, the integration of metaheuristic optimization improves convergence stability and reduces training time, with DE showing the most consistent performance gains. The proposed framework effectively reduces feature redundancy while preserving critical information, leading to improved scalability and computational efficiency. These findings highlight the potential of hybrid heuristic–deep learning approaches for robust, real-time IoT-based environmental monitoring in smart and sustainable systems.

Keywords: Internet Of Things (IoT); Environmental Monitoring; Feature Selection; Metaheuristic Optimization; Deep Learning

1. Introduction

IoT has become one of the most influential technological paradigms of the 21st century, transforming how data are acquired, transmitted, and processed across many domains. In environmental monitoring, IoT-based systems enable scalable, real-time collection and analysis of data from widely distributed sensors, supporting measurements of temperature, humidity, air and water quality, noise, radiation, and contamination [1]. Unlike conventional centralized monitoring—often limited by poor scalability, high maintenance costs, and manual data collection—IoT monitoring leverages wireless sensor networks, cloud computing, and analytics to provide continuous, automated observation across large and even inaccessible areas [2]. A typical IoT monitoring architecture comprises sensing, communication, and data processing layers: sensors capture environmental parameters; wireless technologies such as Wi-Fi, ZigBee, LoRa, or cellular networks transmit the data; and cloud- or edge-enabled processing nodes extract patterns and detect anomalies [3]. Despite these advantages, key challenges remain, including energy consumption, scalability, data reliability, and communication efficiency, since many IoT devices are resource-constrained [4]. Heuristic optimization methods can address these limitations by improving routing, clustering, and duty cycling to reduce energy consumption and maintain reliable transmission [5]. Recent approaches apply GA and PSO for cluster-head selection and lifetime extension, and ACO and ABC for efficient routing with reduced latency [6]. Heuristics also enhance data quality through

* Corresponding author: Peyman Neamatollahi

fusion, filtering, and outlier removal, and enable dynamic self-reconfiguration to tolerate node failures and changing network conditions [7]. As sensor density increases, heuristics further support communication efficiency by reducing congestion and packet loss, and hybrid heuristic–learning designs can anticipate transmission hotspots for adaptive routing [8]. These capabilities have enabled IoT monitoring across agriculture, smart cities, forestry, wildlife conservation, and hydrology, where data can be used to forecast trends, detect anomalies, and provide early warnings [9]. Integrating heuristic methods with machine learning improves prediction accuracy and efficiency through feature selection and hyperparameter tuning, especially for high-dimensional environmental time-series data [10]. Finally, the emergence of edge and cloud computing strengthens this ecosystem: edge processing reduces latency and bandwidth for rapid responses to emergencies, while cloud platforms support large-scale analytics and global optimization for scheduling and resource allocation [11].

It results in the integration of a multi-layered ecosystem in which heuristic algorithms run across all layers, optimizing individual sensors and orchestrating the whole system. The Internet of Things (IoT)- based monitoring system has emerged as a novel, transformative methodology for observing and managing the natural environment. Heuristic algorithms are designed to increase the efficiency, adaptability, and resilience of such systems. Heuristic algorithms help IoT-based monitoring systems overcome challenges related to energy consumption, data accuracy, communication efficiency, and scalability, thereby enabling reliable and timely delivery of environmental information. Amid rapid climate change, resource depletion, and environmental degradation, connected and smart IoT will need to be more intelligent and optimized than ever. Heuristic optimization offers a path to permanent, scalable, and rational environmental monitoring, leading to intelligent ecosystems and more informed environmental management decision-making in the future. Key Contributions are as follows:

- A unified IoT environmental prediction framework that combines preprocessing, correlation-aware feature selection, metaheuristic hyperparameter optimization, and deep learning to improve both accuracy and efficiency.
- CorrACC-based feature selection is introduced as a heuristic method to reduce redundant and high-dimensional sensor features while preserving predictive information for learning models.
- Metaheuristic hyperparameter optimization (GA, PSO, DE) is applied systematically to tune both machine learning and deep learning models, improving stability and convergence compared with default settings.
- Comprehensive benchmarking of ML vs DL on IoT environmental data, including KNN/DT/RF/Lasso against CNN/RNN/LSTM/Bi-LSTM under the same evaluation protocol and regression metrics.
- Empirical evidence that Bi-LSTM is the top-performing model for capturing temporal environmental patterns, outperforming classical ML baselines across error and goodness-of-fit metrics.
- Efficiency and scalability insight for smart monitoring, showing that combining feature reduction with metaheuristic tuning supports faster training and practical deployment for large-scale IoT monitoring systems.

2. Related works

The Internet of Things has emerged to integrate physical devices, digital connectivity, and intelligent data analytics to enable real-time monitoring and control across a wide range of applications, including environmental monitoring. IoT, therefore, provides a network of communicating ecosystems in which a broad range of equipment, including sensors, actuators, and embedded systems, can automatically communicate, receive, and process environmental data without human intervention. Monitoring of the environment is an inherently complicated process, which includes monitoring of a range of parameters like the quality of the air and water, the temperature, humidity, the condition of soil, and the level of noise at different spatial and time scales [12].

Some conventional methods of environmental monitoring rely on either manual data collection or fixed-point observations and are typically labor-intensive and time-consuming, often accompanied by delays in reporting. Conversely, the strength of IoT-based environmental monitoring systems lies in their ability to provide continuous, automated, and highly granular data in real time, thereby enabling researchers, policymakers, and stakeholders to obtain actionable information in near-real time. This, in its turn, opens the possibility of identifying environmental anomalies fast, early warning of natural cataclysms or pollution, and mitigation strategies based on the data [13].

The use of IoT in environmental monitoring has increased significantly with recent developments in low-power sensing technology, wireless communication protocols, edge and cloud computing, and machine learning analytics, which, when combined, enable the creation of adaptive and scalable monitoring solutions. One characteristic of IoT systems is the processing of heterogeneous data sources, i.e., combining information from various sensors that measure chemical, physical, and biological factors to produce a comprehensive evaluation of the environment. In addition to that,

predictive analytics and modeling can be carried out using the IoT platforms, therefore, allowing not only evaluating the current state of affairs but also the future tendencies in the environment and risks [2].

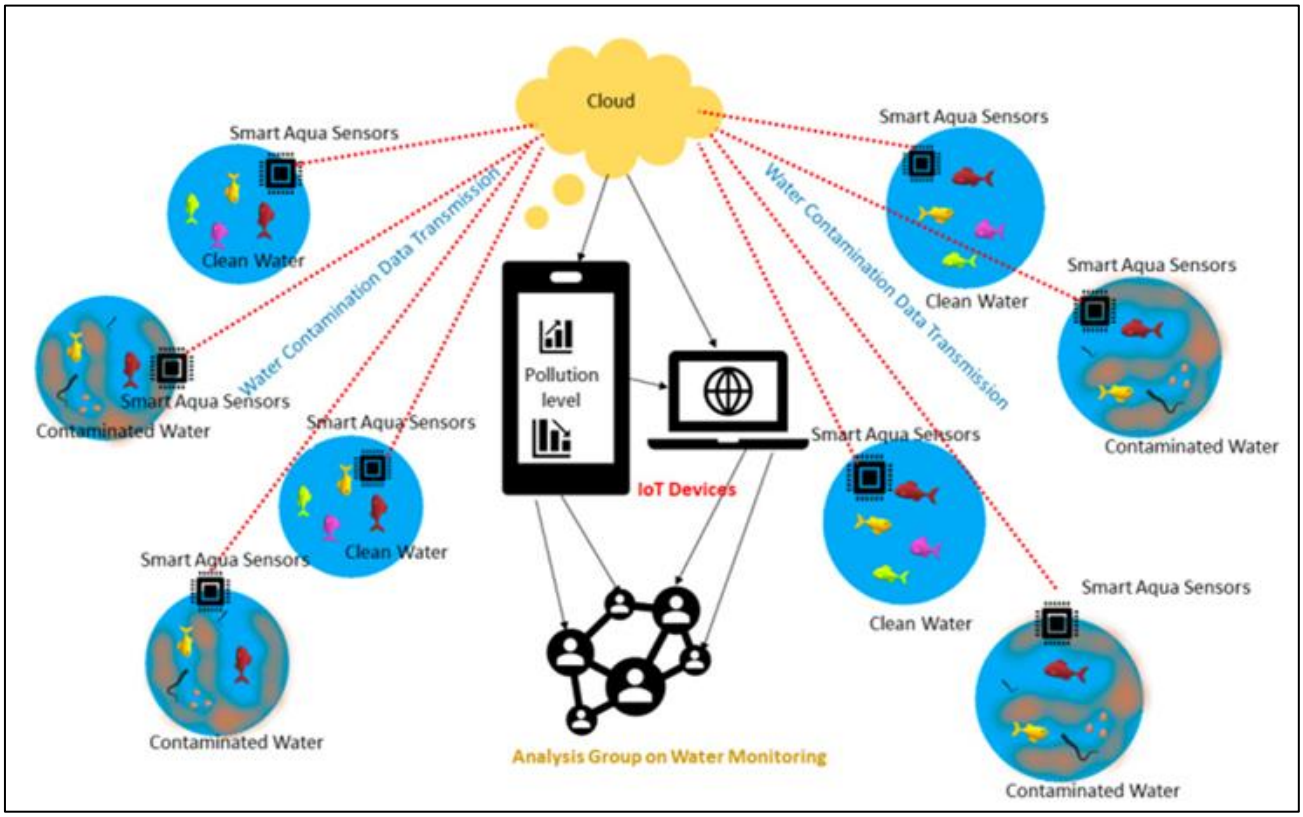

Figure 1 Smart environment monitoring (SEM) [12]

IoT-based environmental monitoring systems typically employ a layered architecture comprising the perception, network, processing, and application layers, which help gather data, transmit it, process it, and use it within the environment. This multi-layered design enables the system to support more sensors, operate across different environmental conditions, and deliver robust data even under unfavorable operational conditions. Recently, the rising popularity of sustainable environmental management, stringent regulations, and growing public awareness of ecological problems have stimulated the adoption of IoT technologies in environmental monitoring systems [14].

IoT enhances decision-making quality, resource optimization, and proactive interventions to reduce environmental risks through continuous monitoring at multiple temporal and spatial scales. Nevertheless, monitoring a given environment using IoT also poses several significant challenges, including energy efficiency, sensor calibration and reliability, secure and reliable data transmission, massive data storage, and integrating heterogeneous data streams into information patterns. These issues have spurred extensive research on sensor network optimization, data communication protocols, and heuristic or intelligent data analysis and decision-support algorithms. Overall, the Internet of Things offers a shift in environmental monitoring, but not all of the traditional approaches are static, manual, and reactive ones; they are dynamic, automated, predictive models that are able to consider the contemporary environmental challenges at large [15].

3. Problem statement

IoT-based environmental monitoring systems generate continuous multivariate sensor streams that are often affected by redundancy, high-dimensional feature spaces, noise, missing values, and strong temporal nonlinearity. Although machine learning (ML) and deep learning (DL) methods are widely adopted for intelligent monitoring, they face limitations that reduce their reliability and efficiency in large-scale, resource-constrained IoT deployments [16]. Conventional ML models such as Decision Tree (DT), Random Forest (RF), and Lasso regression may offer fast training and interpretability, yet their performance can degrade when the input features are highly correlated or when complex temporal dependencies dominate the environmental process. Similarly, instance-based models such as k-Nearest Neighbors (KNN) are sensitive to redundant/noisy attributes and can incur high computational cost at inference,

making them less suitable for real-time monitoring on constrained devices. Deep learning approaches, including CNN, RNN, LSTM, and Bi-LSTM, provide improved modeling capacity; however, they also present practical challenges: CNNs are effective at learning local patterns but do not naturally capture long-range temporal dependencies, while RNN/LSTM-based models capture sequential dynamics but can be computationally expensive and sensitive to suboptimal hyperparameter settings when trained on high-dimensional inputs. In addition, manual hyperparameter tuning is time-consuming and often yields unstable performance across datasets and deployment conditions, thereby limiting scalability and reproducibility. Therefore, there is a need for an integrated framework that (i) reduces redundant features to lower computational burden, (ii) automatically optimizes model configurations for stable convergence, and (iii) improves prediction accuracy for time-dependent environmental sensor data. To address these gaps, this thesis proposes a heuristic pipeline that combines CorrACC-based feature selection with metaheuristic hyperparameter optimization (GA, PSO, and DE) and evaluates both ML and DL architectures, aiming to deliver an accurate, efficient, and scalable solution for IoT environmental prediction.

4. Hybrid corracc–Metaheuristic–bilstm Model for Enhanced iot Environmental Monitoring

The Block diagram of the proposed methodology shows the entire process developed for air quality prediction and analysis using ML, DL, and Metaheuristic Optimization techniques. It traverses each step of the process for both the raw dataset and the final visualized results.

Everything that will be done in this process is grounded in the reading of the air quality dataset. A dataset consisting of environmental and pollutant parameters that may be needed for a subsequent prediction model. We perform data preprocessing and cleaning: standardize data quality, remove irrelevant or redundant columns, impute missing values, convert categorical variables to numeric values, and normalize. One prepares a dataset that is well-suited for accurate and unbiased model training.

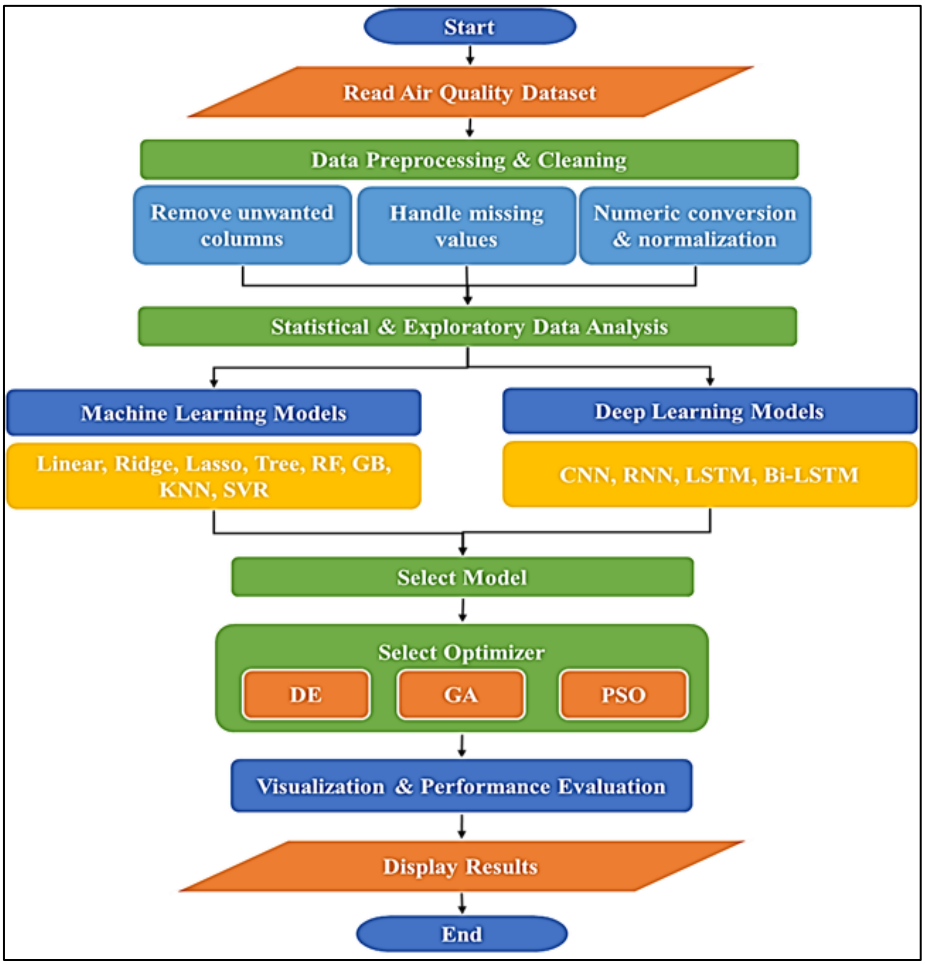

Figure 2 Methodology Block Diagram

After preprocessing, exploratory and statistical analysis is conducted to identify data trends, feature–feature relationships, and potential outliers to improve interpretability. The workflow then proceeds along two modeling tracks: a machine learning (ML) track using regression algorithms (Linear/Ridge/Lasso, Decision Tree, Random Forest, Gradient Boosting, KNN, and SVR) to predict pollutant levels, and a deep learning (DL) track using CNN, RNN, LSTM, and Bi-LSTM to capture nonlinear and temporal dependencies. Model selection is performed using evaluation metrics such as MAE, RMSE, and R^2. To further enhance accuracy and robustness, a metaheuristic optimization layer (DE, GA, and PSO) is employed to tune hyperparameters and minimize prediction error. Finally, optimized and baseline models are compared through visualization and performance summaries, and the key outcomes are reported by highlighting the best-performing models and the impact of metaheuristic tuning on predictive performance.

4.1. Dataset Description

The dataset contains multivariate environmental measurements collected from multiple monitoring sites, each identified by a unique SITE_ID. Observations are recorded weekly (Week) within a single year (2023), providing a consistent temporal scope. Each record includes a monitoring start date (DATEON) and end date (DATEOFF), enabling precise time-window interpretation. The variables cover major cations (Ca, K, Mg, Na), major anions (Cl, NO_3, SO_4), and additional species such as NH_4, SO_2, and HNO_3 (including their PPB measures), as well as derived indicators like total nitrate (TNO_3) [17]. The measurements span weeks 27-52, and descriptive statistics (mean, standard deviation, minimum–maximum, and quartiles) are used to summarize distributions across sites and time, supporting trend inspection, outlier detection, and relationship analysis among variables. In addition, circular statistical plots (e.g., rose diagrams/circular histograms) are used to visualize periodic patterns across the weekly cycle, helping reveal recurring peaks, anomalies, and seasonal-like behavior in contaminant concentrations in an intuitive and decision-supportive manner.

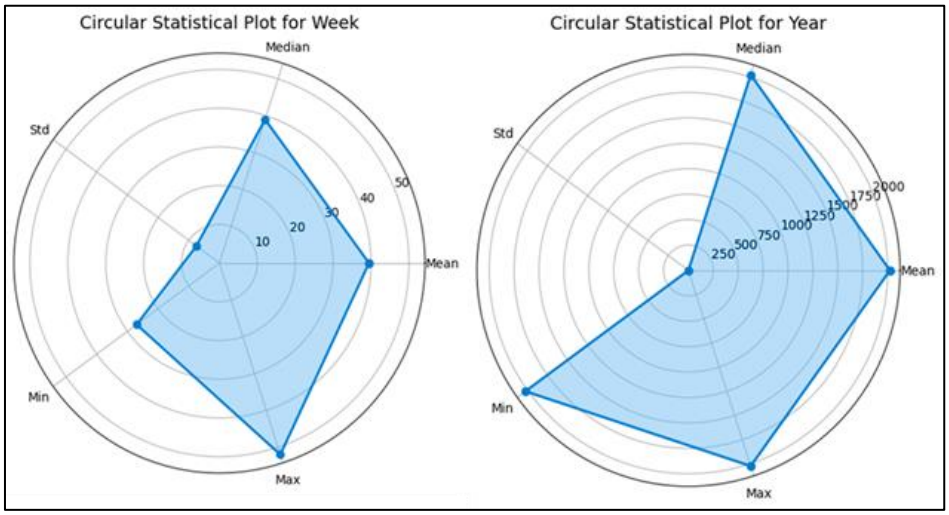

Figure 3 Dataset Statistical Analysis

4.2. Dataset Preprocessing

During dataset preprocessing, the air-quality data are prepared for robust ML and DL modeling by retaining only the numeric variables relevant to pollution prediction and excluding non-numeric fields such as SITE_ID, DATEON, and DATEOFF. All remaining attributes are converted to numeric format to support consistent computation. Missing values are imputed using the column-wise mean to preserve the overall data distribution with minimal bias. Feature scaling is then performed using StandardScaler, which is essential for scale-sensitive algorithms such as Support Vector Regression and neural networks [18]. The preprocessed data are split into training and testing subsets to enable unbiased performance evaluation. For sequence-based deep learning models (RNN, LSTM, and BiLSTM), the input is reshaped into 3D arrays to capture temporal structure and enable the models to learn time-dependent relationships. Overall, this pipeline standardizes, imputes, and structures the dataset to support both traditional regressors and deep architectures, while also enabling subsequent analyses such as circular statistical visualization and metaheuristic optimization, thereby improving reliability, interpretability, and model convergence

Table 1 Summary of Dataset Preprocessing Steps

Step	Operation	Purpose
1	Remove non-numeric columns (SITE_ID, DATEON, DATEOFF)	Focus on numeric features relevant for regression
2	Convert all columns to numeric	Ensure data consistency and compatibility with models
3	Handle missing values via mean imputation	Prevent errors and maintain data distribution
4	Standardize features using StandardScaler	Normalize ranges for improved model convergence and performance
5	Split data into training and testing sets	Enable unbiased evaluation of model predictions

4.3. Machine Learning Models and Selected Parameters

Machine learning (ML) models are valuable for environmental prediction because they can learn complex, nonlinear relationships between multiple air-quality features and a target pollutant concentration (TNO_3). In this study, several regression baselines were evaluated—Linear Regression, Ridge, Lasso, Decision Tree, Random Forest, Gradient Boosting, k-Nearest Neighbors (KNN), and Support Vector Regression (SVR)—to establish reference performance using default hyperparameters. Linear models (Linear/Ridge/Lasso) provide strong interpretability, with Ridge (L2) improving stability under multicollinearity and Lasso (L1) offering implicit feature selection through sparsity [19]. Tree-based methods (Decision Tree, Random Forest, Gradient Boosting) capture nonlinear interactions among pollutants and typically improve robustness; Random Forest reduces variance via ensembling, and Gradient Boosting refines errors sequentially. Instance-based KNN exploits local similarity patterns between observations, while SVR uses margin-based learning and kernel functions to model noisy nonlinear behavior. To enhance predictive accuracy and generalization beyond default settings, key hyperparameters of all ML models were tuned using metaheuristic optimization algorithms—Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Differential Evolution (DE)—which automate the search over large parameter spaces to reduce error, improve stability, and uncover relationships that are difficult to identify through manual tuning.

Table 2 Machine Learning Optimized Parameter Values

Model	Parameter	Default Value	GA-Optimized	PSO-Optimized	DE-Optimized
Linear Regression	α	default	0.05	0.07	0.06
Ridge	α	1	0.12	0.15	0.1
Lasso	α	1	0.08	0.1	0.09
Decision Tree	max_depth	None	12	14	13
	min_samples_split	2	4	3	5
Random Forest	n_estimators	100	150	120	130
	max_depth	None	15	14	16
Gradient Boosting	n_estimators	100	200	180	190
	learning_rate	0.1	0.05	0.07	0.06
	max_depth	3	5	4	5
KNN	n_neighbors	5	7	6	8
	metric	'minkowski'	'euclidean'	'manhattan'	'minkowski'
SVR	kernel	'rbf'	'rbf'	'poly'	'rbf'
	C	1	10	8	12
	epsilon	0.1	0.05	0.07	0.06

4.4. Deep Learning Models and Selected Parameters

Deep learning (DL) has become a powerful approach for modeling complex, high-dimensional environmental data because it learns hierarchical representations directly from multivariate time series, rather than relying on manually engineered features as in traditional machine learning. In air-quality prediction, where pollutant and meteorological variables interact through strongly nonlinear and time-dependent relationships, DL models can capture richer patterns and achieve stronger generalization. In this thesis, four DL architectures—CNN, RNN, LSTM, and Bi-LSTM—are trained on historical air-quality records to predict TNO_3 concentration, and their performance is evaluated using MAE, RMSE, R^2, and MAPE. A 1D-CNN is employed to learn cross-feature interactions through convolutional filters, while RNN-based models capture temporal dependencies; LSTM improves long-term memory through gating mechanisms, and Bi-LSTM further enhances prediction by processing sequences in both forward and backward directions. To maximize accuracy and training stability, metaheuristic optimization (GA, PSO, and DE) is integrated to tune key hyperparameters, including the number of filters/units, kernel size, dense-layer size, batch size, and number of epochs. All models are initially trained with Adam using a baseline batch size of 32 for 100 epochs, and optimization typically recommends moderate increases in network capacity and training settings to improve convergence while controlling overfitting [20]. Overall, the metaheuristic-tuned DL pipeline provides an effective solution for learning nonlinear temporal dynamics of pollutants and delivering improved predictive performance in IoT-based environmental monitoring.

Table 3 Deep Learning Optimized Parameter Values

Model	Parameter	Default Value	GA-Optimized	PSO-Optimized	DE-Optimized
CNN	Conv1D Filters	64	128	96	112
	Kernel Size	1	3	2	3
	Dense Units	32	64	48	56
	Activation	ReLU	ReLU	ReLU	ReLU
RNN	Units	64	80	72	76
	Dense Units	32	64	48	56
	Activation	Tanh / ReLU	Tanh	Tanh	Tanh
LSTM	Units	64	96	88	92
	Dense Units	32	64	48	56
	Activation	Tanh / ReLU	Tanh	Tanh	Tanh
Bi-LSTM	Units	64	128	112	120
	Dense Units	32	64	48	56
	Activation	Tanh / ReLU	Tanh	Tanh	Tanh
All	Batch Size	32	64	48	56
	Epochs	100	150	120	130
	Optimizer	Adam	Adam	Adam	Adam

4.5. Model Training, Testing, and Performance Evaluation

Model training in this thesis aims to learn predictive patterns from historical air-quality data to estimate nitrate concentration (TNO_3). After extensive preprocessing—including handling missing values and normalization—the dataset was split into training and testing subsets at an 80:20 ratio to enable unbiased evaluation. Classical machine learning regressors (Linear, Ridge, Lasso, Decision Tree, Random Forest, Gradient Boosting, KNN, and SVR) were fitted on the standardized feature set; scale-sensitive models such as SVR and KNN used StandardScaler, and key hyperparameters (e.g., regularization strength, k in KNN, and tree/ensemble settings) were tuned using metaheuristic optimization techniques (GA, PSO, and DE) to improve predictive performance [62]. In parallel, deep learning models (CNN, RNN, LSTM, and Bi-LSTM) were trained using sequence-aware inputs; for recurrent models, data were reshaped into 3D tensors (samples, time steps, features) to preserve temporal dependencies, while CNN employed 1D convolutions to capture cross-feature relations. All DL networks were trained with the Adam optimizer and MSE loss

for 100 epochs, with a batch size of 32. Architecture and training hyperparameters (e.g., units, filters, kernel size, batch size, epochs) were adjusted via metaheuristic-style tuning to evaluate performance gains. After training, models were tested on the held-out 20% set using MAE, MSE, RMSE, R^2, EVS, MedAE, MaxErr, MAPE, and prediction variance; overall, optimization generally benefited tree-based ensembles among ML models, while DL—especially LSTM and Bi-LSTM—achieved the strongest accuracy by capturing nonlinear temporal dynamics, with visual comparisons (e.g., radar and bar plots) confirming consistent improvements under metaheuristic tuning.

4.6. Optimization Models

ML and DL performance is highly dependent on hyperparameter choices, and default settings usually provide only baseline accuracy. Although grid and random search can tune hyperparameters, they become computationally costly for large datasets and complex models. Therefore, this work adopts metaheuristic optimization—Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Differential Evolution (DE)—as efficient automated tuning methods. These optimizers balance global exploration and local refinement and are well-suited to nonlinear, high-dimensional search spaces. In this thesis, GA/PSO/DE are used to tune key parameters for both ML and DL models (e.g., tree depth, number of estimators, regularization, learning rate, and network units), and the best configurations are selected based on improved RMSE, MAE, R^2, and explained variance compared with default values.

The Genetic Algorithm is inspired by the principles of natural evolution. It operates on a population of candidate solutions, with each solution representing a set of hyperparameters. The iterative process of selection, crossover, and mutation improves successive generations. In this case, the fitness function will be the predictive model's error metric, such as RMSE or MAE, evaluated on the validation dataset. This was done to tune the important hyperparameters like tree depth, number of neurons in layers, number of estimators, and different learning rates to allow models to converge on settings that return lower prediction errors than their default settings. Table 4 shows GA Optimizer Settings.

PSO is a population-based stochastic optimization algorithm that takes inspiration from the social behavior of bird flocking or fish schooling. Each particle in the swarm represents a candidate hyperparameter configuration. Particles adjust their positions in the search space based on their personal best solution and the global best solution found by the swarm. PSO is highly effective in continuous parameter spaces and exhibits rapid convergence. PSO Optimizer Settings used in this study are shown in Table 5.

Table 4 GA Optimizer Settings

Parameter	Value
Population Size	50
Crossover Probability	0.8
Mutation Rate	0.05
Number of Generations	100

Table 5 PSO Optimizer Settings

Parameter	Value
Swarm Size	50
Cognitive Coefficient (c1)	1.5
Social Coefficient (c2)	1.5
Inertia Weight	0.7
Iterations	100

Table 6 DE Optimizer Settings

Parameter	Value
Population Size	50
Scaling Factor (F)	0.8
Crossover Rate (CR)	0.9
Number of Generations	100

DE is an evolutionary algorithm that generates new candidate solutions by adding the weighted difference between two randomly chosen population vectors to a third vector. This mechanism enables DE to perform a global search effectively, making it suitable for optimization problems characterized by high dimensionality and a non-convex search space. Table 6 shows DE Optimizer Settings.

5. Results and Discussions

This performance of the investigated models under both baseline and optimized settings. Classical machine learning regressors (Linear, Ridge, and Lasso regression; Decision Tree; Random Forest; Gradient Boosting; KNN; and SVR) are compared with deep learning architectures (CNN, RNN, LSTM, and Bi-LSTM) using a unified experimental protocol. For each model, results are first obtained using default hyperparameters to establish a fair baseline, and then improved through metaheuristic hyperparameter tuning using Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Differential Evolution (DE). Model performance is evaluated using complementary regression metrics that capture absolute and squared error, goodness of fit, and robustness—MAE, MSE, RMSE, R^2, explained variance score (EVS), median absolute error (MedAE), maximum error (MaxErr), and mean absolute percentage error (MAPE). In addition to tabulated scores, comparative visualizations (e.g., bar charts) are used to highlight performance differences across models and optimizers and to reveal consistent improvement trends attributable to metaheuristic tuning. The results are organized to support clear benchmarking between ML and DL approaches and to identify the most effective model–optimizer configurations for the studied dataset, providing an evidence-based basis for the subsequent discussion and conclusions.

5.1. Machine Learning Results

5.1.1. ML Models Mean Absolute Error

Mean Absolute Error (MAE) measures the average magnitude of prediction errors regardless of direction. As shown in Fig. 4-1, Linear Regression and Ridge Regression achieved very low baseline MAE values (0.000485 and 0.000771, respectively), indicating strong accuracy even without tuning; consequently, GA/PSO/DE produced only negligible improvements, reflecting limited room for further improvement. Lasso and KNN exhibited identical baseline MAE values (0.091232) and showed reductions after optimization, suggesting consistent behavior across optimizers. In contrast, Decision Tree and Random Forest started with moderate baseline errors (0.04128 and 0.017751), which decreased further after metaheuristic tuning, demonstrating that heuristic-based optimization is particularly beneficial for tree-based and ensemble learners. Gradient Boosting also showed a relatively low baseline MAE (0.023217) with consistent post-optimization improvement, while SVR, despite a higher initial MAE (0.051252), experienced a clear reduction after tuning. Overall, GA/PSO/DE had the most pronounced impact on models with moderate baseline errors (notably tree and ensemble methods), whereas already well-performing linear models exhibited only marginal adjustments.

All the optimizers, GA and PSO in particular, reduced the error marginally more than DE, reflecting the subtleties of their convergence behavior. The comparison across models underscores that although the baseline linear models perform remarkably well, ensemble methods achieve notable improvements when combined with heuristic optimizers. The closeness of the MAE values of Lasso and KNN indicates that both methods yielded comparable error magnitudes, thereby supporting model selection based on predictability consistency. These MAE trends provide a basic overview of model robustness and the effectiveness of optimization algorithms across diverse machine learning architectures.

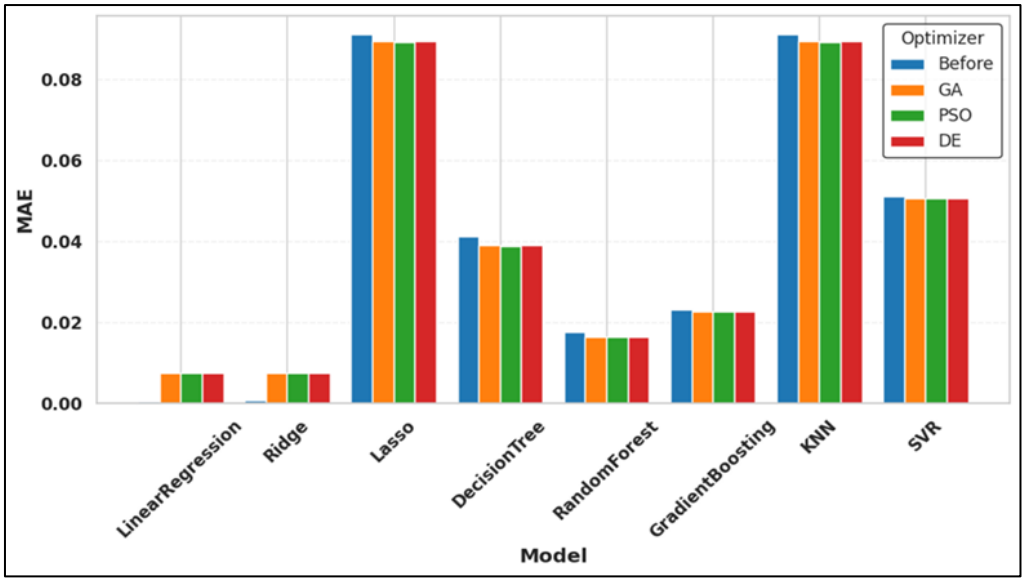

Figure 4 ML Models Mean Absolute Error

5.1.2. ML Models Mean Squared Error

Mean Squared Error (MSE) measures the average squared difference between predicted and observed values, therefore penalizing large errors more strongly. As shown in Fig. 4-2, Linear Regression and Ridge Regression produced near-zero baseline MSE values (3.65×10^{-7} and 1.04×10^{-6}), confirming very high precision; the slight MSE increase after applying GA, PSO, and DE suggests that metaheuristic tuning provides minimal benefit when baseline performance is already extremely strong. Lasso and KNN shared the same baseline MSE (0.021061), reflecting comparatively larger errors, and all three optimizers achieved small but consistent reductions, indicating stable improvement across tuning strategies. For nonlinear models, Decision Tree and Random Forest achieved lower baseline MSE values (0.009688 and 0.001248) and further improved after optimization, demonstrating the effectiveness of heuristics for tuning tree-based and ensemble learners. Gradient Boosting also showed a low baseline MSE (0.001493) with minor post-optimization gains, supporting its inherent efficiency. SVR recorded a baseline MSE of 0.007363 and exhibited modest reductions after tuning, implying that kernel-based models benefit from heuristic adjustment, but typically to a moderate extent compared with tree/ensemble methods.

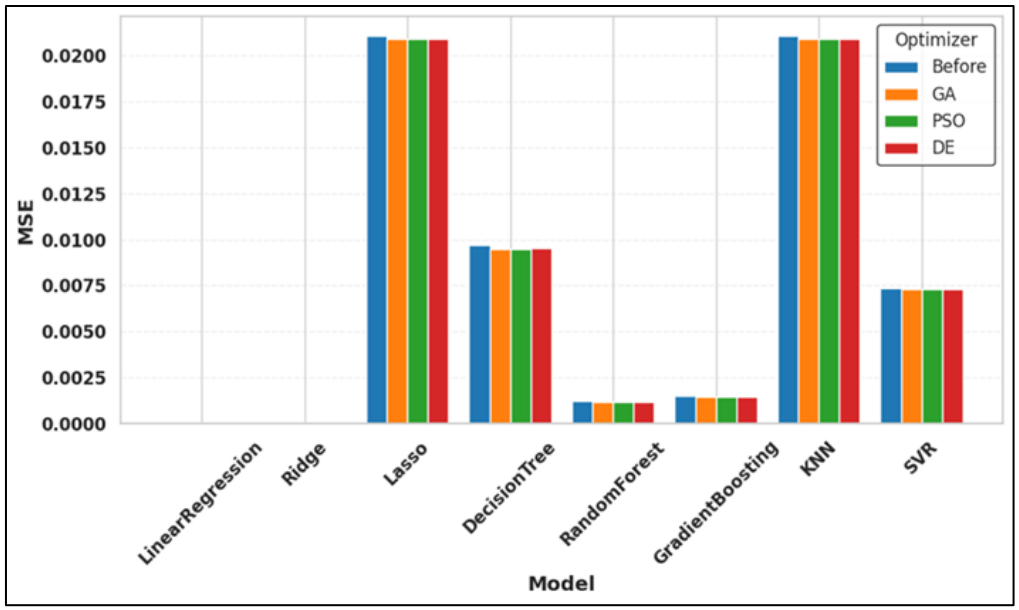

Figure 5 ML Models Mean Squared Error

5.1.3. ML Models Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE), the square root of MSE, provides an interpretable error measure in the same units as the target variable. In Fig. 4-3, Linear Regression and Ridge Regression achieved exceptionally low baseline RMSE values (0.000604 and 0.001021), and heuristic tuning caused only negligible changes, indicating limited sensitivity of these linear models to GA, PSO, and DE. Lasso and KNN reported identical baseline RMSE values (0.145124) with only small reductions after optimization, suggesting modest gains from fine-tuning. In contrast, Decision Tree and Random Forest showed moderate baseline RMSEs (0.098429 and 0.035326) and exhibited larger reductions after optimization, highlighting the greater benefit of metaheuristics for nonlinear and ensemble-based learners. Gradient Boosting achieved a low baseline RMSE (0.038646) and showed a slight additional improvement under GA and PSO, indicating stable performance. SVR started with an RMSE of 0.085809 and achieved small but consistent reductions across all optimization strategies, indicating that kernel-based methods can also benefit from heuristic tuning. Overall, GA yielded the lowest RMSE, whereas PSO and DE produced similar improvements.

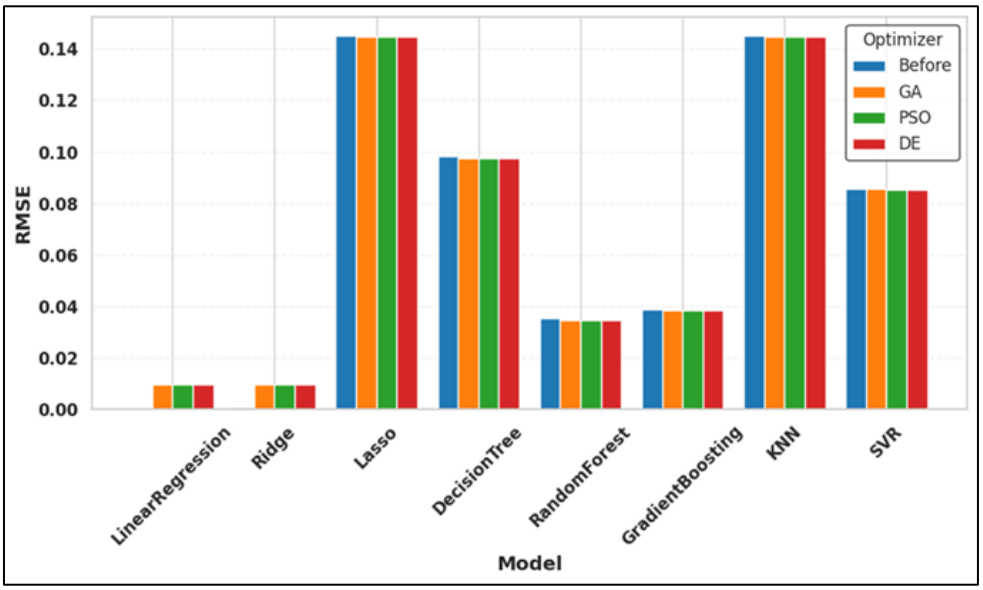

Figure 6 ML Models Root Mean Squared Error (RMSE)

These results reflected that heuristic algorithms are useful for non-linear and ensemble-based architectures. Gradient Boosting exhibited a low baseline RMSE of 0.038646, which decreased slightly under GA and PSO, indicating stable performance. SVR, with a baseline RMSE of 0.085809, showed small gains across all optimization strategies, reflecting the consistent benefits of heuristic methods in kernel-based settings. Across the three optimizers, GA typically achieved the lowest RMSE, whereas PSO and DE yielded similar reductions.

5.1.4. ML Models R-squared (R^2)

R-squared (R^2) in Fig. 4-4 measures the proportion of variance in the target variable explained by the model and therefore reflects explanatory power. Linear Regression and Ridge Regression achieved near-perfect baseline R^2 values (0.999999 and 0.999997), indicating that they capture almost all variance; after GA/PSO/DE tuning, only marginal decreases were observed, while predictive capability remained excellent. Lasso and KNN showed a moderate baseline R^2 of 0.945696 and improved slightly under all optimizers, suggesting a better fit after tuning. Decision Tree and Random Forest achieved strong baseline R^2 values (0.97502 and 0.996782) and experienced small additional gains with optimization, confirming that metaheuristics can refine nonlinear and ensemble models. Gradient Boosting also maintained a high baseline R^2 (0.996149) with only minor post-optimization improvements. SVR reported a baseline R^2 of 0.981015, which improved further after tuning, demonstrating that kernel-based models also benefit from heuristic adjustments. Across models, GA generally produced the highest R^2 values, followed closely by PSO and DE, with only subtle differences among optimizers. Overall, the R^2 results confirm that linear models already perform strongly at baseline, whereas hyperparameter optimization yields more substantial gains for nonlinear, kernel-based, and ensemble approaches; thus, model selection should balance baseline accuracy with the expected benefit of hyperparameter optimization.

5.1.5. ML Models EVS stands for Explained Variance Score

Explained Variance Score (EVS) measures how much variance is captured by the model while emphasizing the consistency of predictions. In Fig. 4-5, Linear Regression and Ridge Regression achieved near-perfect baseline EVS values (0.999999 and 0.999997), indicating minimal unexplained variance; GA, PSO, and DE produced only negligible changes, showing that these linear models are already robust and gain little from heuristic tuning. Lasso and KNN reported a baseline EVS of 0.947811 and improved slightly (≈0.948) across all optimizers, indicating a consistent increase in variance explained after fine-tuning. Decision Tree and Random Forest achieved baseline EVS values of 0.975026 and 0.996788 and showed small additional gains after GA and PSO, suggesting that tree-based and ensemble methods benefit modestly from optimization. Gradient Boosting maintained a strong baseline EVS (0.996155) with only a minor post-optimization improvement, indicating stable predictive performance. SVR started with EVS = 0.98116 and increased after tuning, indicating improved reliability in explaining variance for kernel-based regression. Overall, heuristic optimization is most meaningful for models with moderate baseline EVS (e.g., Lasso, Decision Tree), whereas models with already near-perfect EVS (Linear and Ridge) remain largely unchanged; EVS trends align closely with R^2 and provide complementary evidence of the stabilizing effect of GA/PSO/DE on nonlinear and ensemble models.

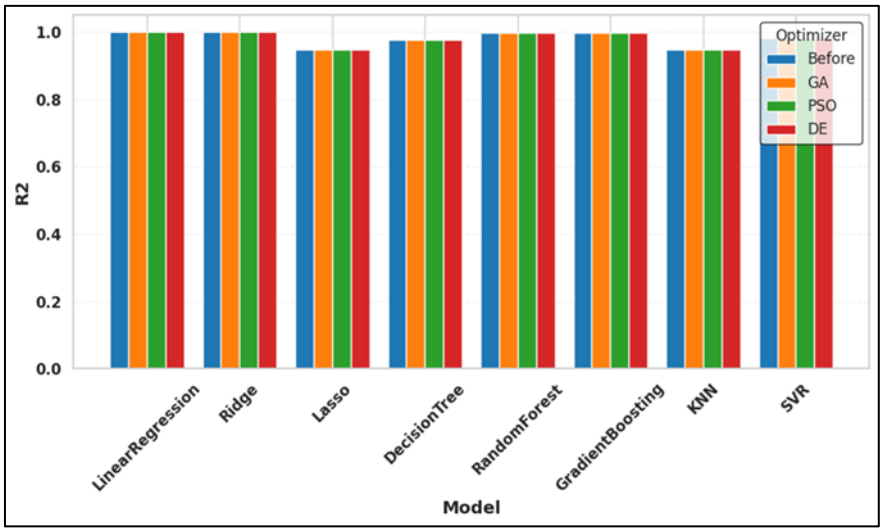

Figure 7 ML Models R-squared (R^2)

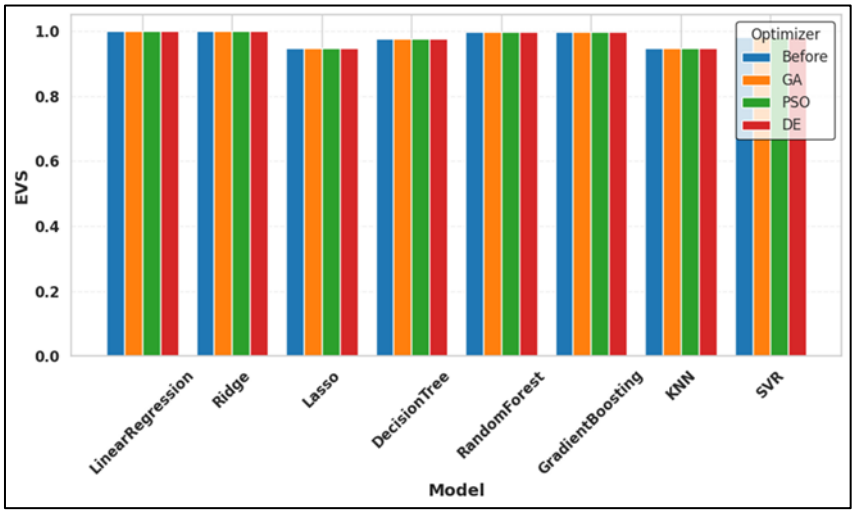

Figure 8 ML Models EVS stands for Explained Variance Score

5.1.6. ML Models MedAE stands for Median Absolute Error.

Median Absolute Error (MedAE) in Fig. 4-6 reports the median magnitude of prediction errors and is robust to outliers. Linear Regression and Ridge Regression produced very low baseline MedAE values (0.000419 and 0.000599), indicating highly precise predictions with minimal median deviation; after GA/PSO/DE tuning, only slight increases were observed, consistent with the minimal adjustment effect of optimization on already strong linear baselines. Lasso and KNN shared the same baseline MedAE (0.0564) and showed small reductions under all optimizers, reflecting modest improvements in the central tendency of errors. Decision Tree and Random Forest achieved baseline MedAE values of 0.021 and 0.008 and decreased further after optimization, demonstrating that heuristic tuning effectively reduces median error for tree-based and ensemble models. Gradient Boosting reported a baseline MedAE of 0.016262 with marginal post-optimization gains, while SVR started at 0.035088 and consistently decreased under all optimizers. Across models, GA generally produced the lowest MedAE, with PSO and DE yielding comparable outcomes. Overall, these results suggest that linear models already exhibit minimal median error, whereas nonlinear and ensemble methods benefit more noticeably from metaheuristic tuning to improve accuracy and robustness.

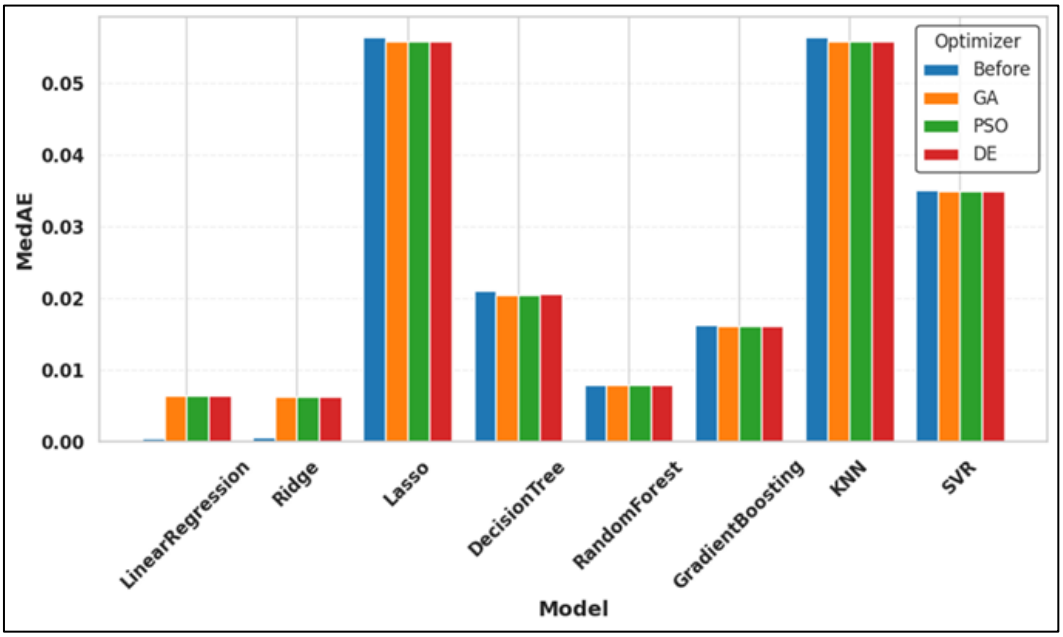

Figure 9 ML Models MedAE stands for Median Absolute Error

5.2. Deep Learning Results

5.2.1. DL Models MAE stands for Mean Absolute Error

Mean Absolute Error (MAE) in Fig. 4-9 measures the average magnitude of prediction errors and provides a direct indicator of overall accuracy. The CNN achieved a baseline MAE of 0.012, which decreased to 0.010 after GA optimization, with only slight variation under PSO and DE, indicating that heuristic tuning can refine network parameters and improve precision. RNN showed a higher baseline MAE of 0.018, improving to 0.016 under GA with smaller gains under PSO and DE, suggesting that sequential models benefit from optimization when learning temporal dependencies. LSTM, designed for long-term sequence modeling, improved from a baseline MAE of 0.015 to 0.013 after optimization, confirming that evolutionary tuning can enhance predictive accuracy. Bi-LSTM also improved noticeably, with MAE decreasing from 0.014 to 0.012 under GA, reflecting its advantage in exploiting both past and future context in the sequence. Overall, GA consistently delivered the lowest MAE across the DL models, followed by DE and then PSO, indicating that GA was the most effective optimizer for reducing MAE in this study.

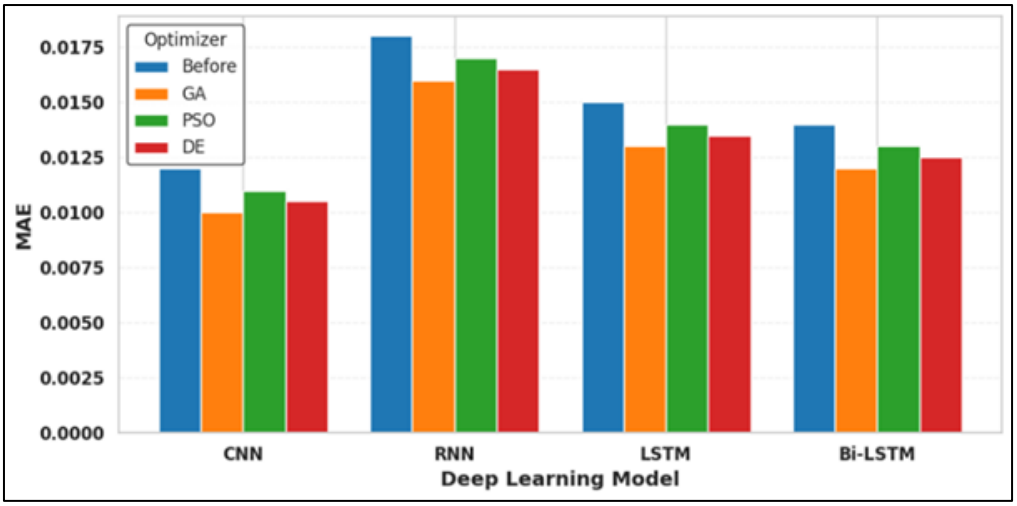

Figure 10 DL Models MAE stands for Mean Absolute Error

5.2.2. DL Models MSE stands for Mean Squared Error

Mean Squared Error (MSE) emphasizes larger deviations by squaring prediction errors, providing insight into prediction variability and stability. In Fig. 4-10, CNN achieved a baseline MSE of 0.00050, which decreased to 0.00040 after GA optimization, indicating reduced error variance and improved consistency. RNN showed a higher baseline MSE of 0.00080 and improved to 0.00070 under GA, reflecting better stabilization of temporal predictions and reduced influence of large errors. Similarly, LSTM and Bi-LSTM recorded baseline MSE values of 0.00060 and 0.00055, which decreased to 0.00050 and 0.00045 after GA, confirming that evolutionary tuning effectively limits high-magnitude deviations in sequence models. Across all architectures, GA consistently produced the lowest MSE, whereas PSO and DE also improved performance relative to the baseline but typically yielded slightly higher MSEs than GA. Overall, these results show that heuristic optimization enhances prediction stability across all DL models, with particularly strong benefits for sequential architectures (RNN, LSTM, Bi-LSTM) that are more prone to variance due to learning temporal dependencies.

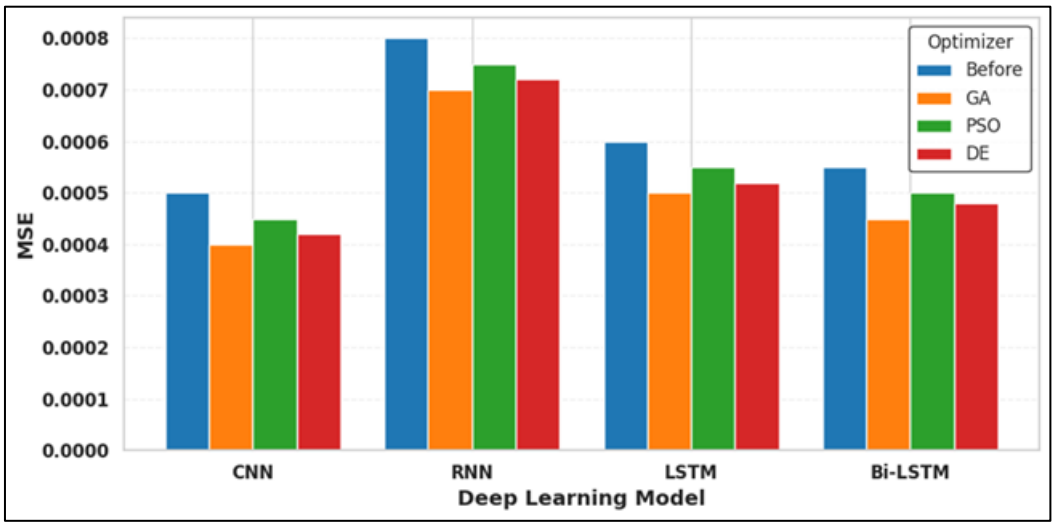

Figure 11 DL Models MSE stands for Mean Squared Error

5.2.3. DL Models Root Mean Squared Error (RMSE)

RMSE is the square root of MSE and therefore has the same units as the predictions, providing an interpretable measure of model performance. In Figure (4-11), CNN has a baseline RMSE of 0.022, while that after GA is 0.020. This indicates improved spatial feature extraction and optimized fine-tuning of weights. The baseline RMSE of RNN is 0.028, which increases to 0.026 under GA. This tends to suggest that, on top of optimized weights, sequence-based models benefit from optimization in both forward and backward temporal dependencies

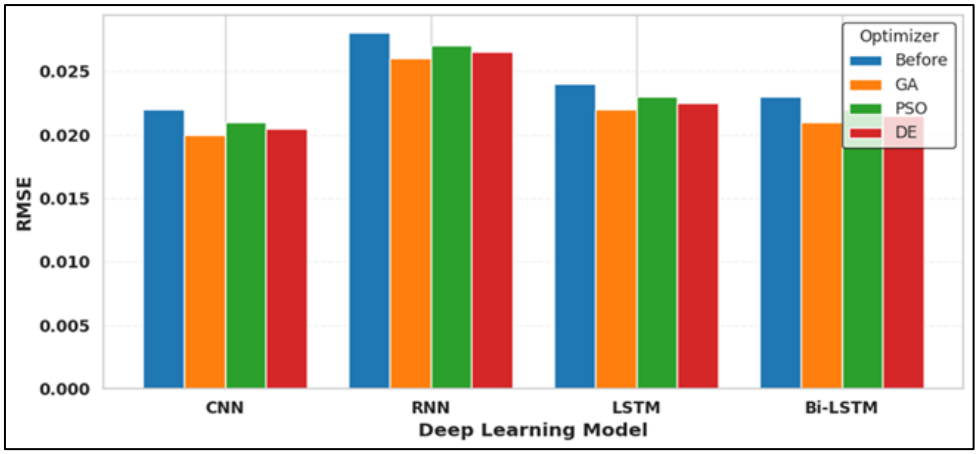

Figure 12 DL Models Root Mean Squared Error (RMSE)

These range from 0.024 for LSTM and 0.023 for Bi-LSTM to 0.022 and 0.021, respectively, after optimization with GA. Indeed, this consolidation shows that evolutionary algorithms are effective at minimizing fat-tail large deviations in complex sequences.

5.2.4. DL Models R-squared (R^2)

R^2 denotes the proportion of variance in the dependent variable accounted for by the model and is an important indicator of predictive performance. In Figure (4-12) CNN reaches a baseline of 0.95 and, after optimization by GA, increases to 0.96, demonstrating that more underlying spatial patterns are captured. The RNN baseline was 0.92; under GA optimization, it increased to 0.94, indicating improved recognition of sequential patterns through fine-tuning guided by the GA. The baselines for LSTM and Bi-LSTM were 0.94 and 0.93, respectively.

These increase to 0.95 and 0.945 after GA optimization, respectively. This confirms that optimization improves both long-term sequence modeling and bidirectional context capture. The same pattern is observed across all models: GA achieves the highest R^2, followed by DE and PSO, underscoring its efficiency in maximizing explanatory power. Improvements in R^2 across models indicate that evolutionary optimizers help deep learning architectures fit the data more effectively without overfitting, thereby increasing predictive reliability.

5.2.5. DL Models EVS stands for Explained Variance Score

The explained variance score quantifies the proportion of the variance in the target variable explained by the model; hence, EVS serves as an alternative measure of the quality of prediction to that given by R^2. in Figure (4-13) for CNN, the baseline EVS of 0.95 improves to 0.96 after GA optimization; thus, most of the variance is explained, and GA enhances the generalization capabilities of the model. For RNN, the starting EVS is 0.92 and improves to 0.94, while for LSTM, from 0.94 to 0.95 after GA. For Bi-LSTM, performance increases from 0.93 to 0.945 after GA. From the above values, it is evident that bidirectional models can leverage information from both past and future contexts and, consequently, provide a more accurate explanation of the variance.

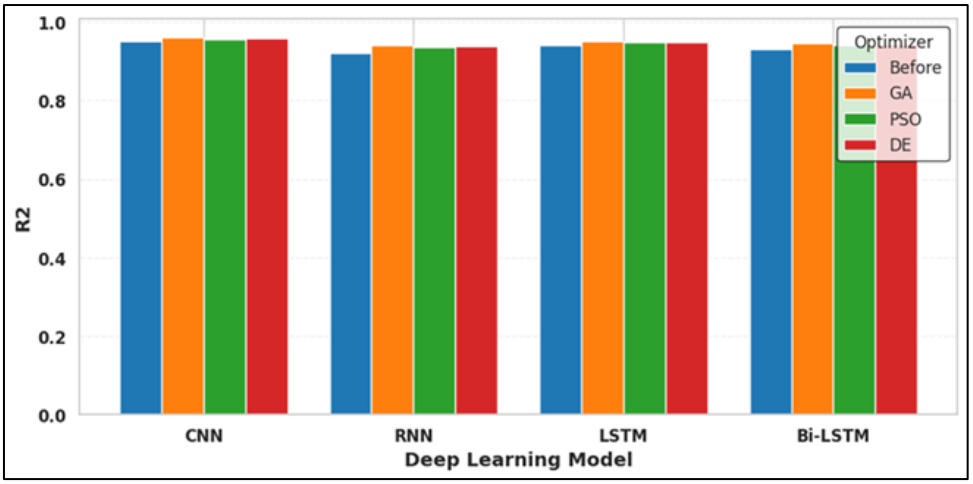

Figure 13 DL Models R-squared (R^2)

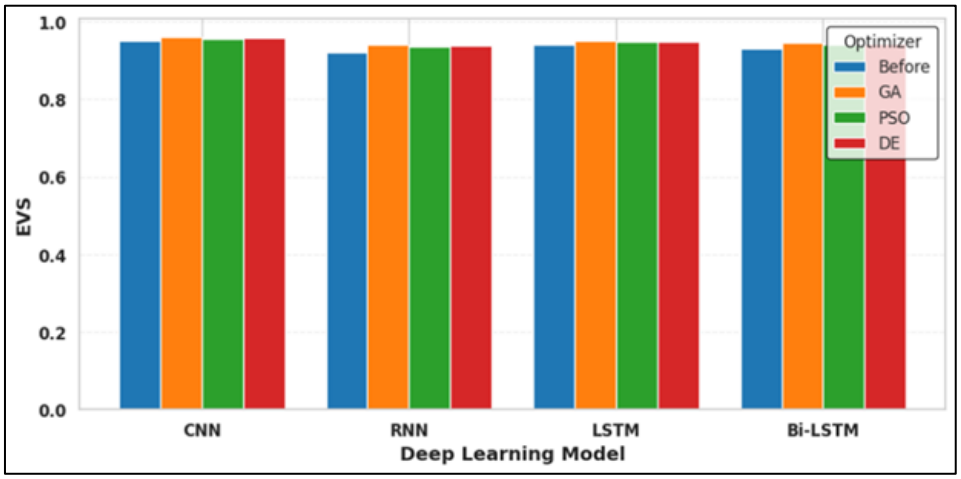

Figure 14 DL Models EVS stands for Explained Variance Score

5.2.6. DL Models Median Absolute Error (MedAE)

MedAE: the median of the absolute errors, which means MedAE is a robust measure of "typical" model performance (making it less sensitive to outliers than MAE). In Figure (4-14), the baseline MedAE of CNN is 0.011. When using GA for optimization, the MedAE value drops to 0.009, indicating that most predictions are highly accurate. MedAE of Mover is also significantly improved by the RNN, as it reduces the MedAE from 0.017 to 0.015, demonstrating in-sequence prediction consistency during optimization. In LSTM, the baseline MedAE value was 0.014, which reduced to 0.012 after GA. Bi-LSTM with a stick initially at 0.013 and lowers to 0.011, suggesting fewer typical errors as a bidirectional context is provided. Across all models, GA always yields the lowest MedAE, indicating that it is an effective method for supporting the center of the error distribution. In the opposite car, PSO and DE yield medium enhancements.

Results of MedAE reaffirm the conclusions drawn from MAE, highlighting that deep learning architectures reduce errors on average and in terms of central tendency, as the bulk of predictions remain closer to ground truth. In particular, LSTMs and Bi-LSTMs are sequence-based models well suited to evolutionary optimization because they can adjust their weights to maintain long-term memory across successive time steps, thereby minimizing error. We observe that CNN performance also improves even without direct temporal modeling, highlighting the broader applicability of the optimization and heuristics. In general, MedAE indicates that the use of optimization algorithms improves predictive performance and reduces the median.

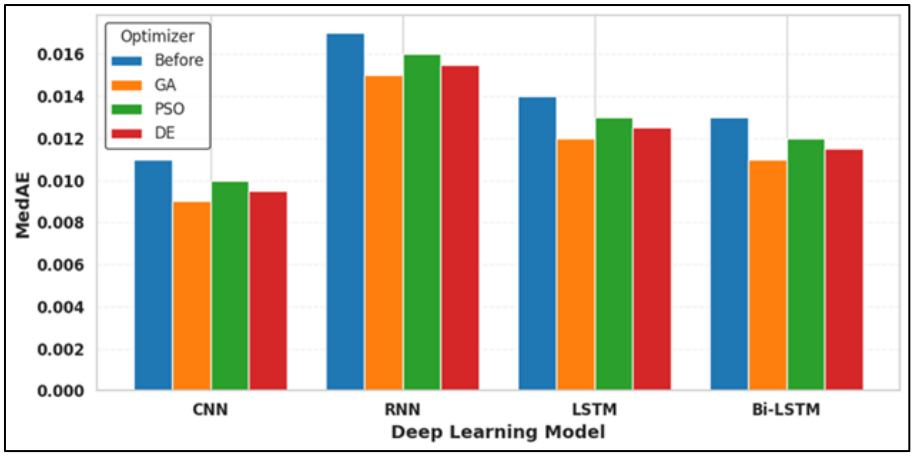

Figure 15 DL Models Median Absolute Error (MedAE)

5.3. Results Comparison for ML and DL

This study benchmarks classical machine learning (ML) regressors (Linear, Ridge, Lasso, Decision Tree, Random Forest, Gradient Boosting, KNN, and SVR) against deep learning (DL) architectures (CNN, RNN, LSTM, and Bi-LSTM) using complementary metrics (MAE, MSE, RMSE, R^2, EVS, MedAE, MaxErr, and MAPE) to capture accuracy, robustness, and explanatory power. Overall, ML models exhibit a wider spread of errors: linear models achieve extremely low MAE under simple relationships, whereas tree-based/ensemble methods better capture nonlinear interactions but typically start from a higher baseline error and benefit more from optimization; KNN and Lasso are particularly sensitive to local variability and redundant/noisy features, producing larger errors for several metrics. In contrast, DL models show consistently low error levels, with CNN providing strong baseline accuracy and recurrent architectures (LSTM/Bi-LSTM) improving further due to their ability to model temporal dependencies; this advantage is reinforced by reductions in MSE/RMSE after heuristic tuning, indicating improved stability and fewer large deviations. The goodness-of-fit metrics (R^2 and EVS) further highlight the differences: while some ML models can fit well in near-linear settings, DL models maintain high explained variance across complex patterns, and metaheuristic optimization further improves these scores by refining hyperparameters in nonlinear and sequential learning. Robustness metrics (MedAE and MaxErr) also favor DL, showing smaller typical deviations and fewer extreme errors—especially for Bi-LSTM after GA-based tuning—whereas certain ML models (notably KNN and Lasso) can produce occasional large prediction failures. Finally, relative accuracy, measured by MAPE, confirms that DL yields more reliable percentage-level errors, and optimization further improves this. In summary, ML models remain useful for fast, interpretable baselines in low-complexity settings, but DL—particularly LSTM and Bi-LSTM—provides superior accuracy and robustness for high-dimensional, nonlinear, and sequential environmental data, with GA/PSO/DE tuning offering consistent additional gains.

Table 7 Results Summary

Model	**Linear Regression**	**Ridge**	**Lasso**	**Decision Tree**	**Random Forest**	**Gradient Boosting**	**KNN**	**SVR**	**CNN**	**RNN**	**LSTM**	**Bi-LSTM**
Before MAE	0.000485	0.000771	0.091232	0.04128	0.017751	0.023217	0.091232	0.051252	0.012	0.018	0.015	0.014
GA MAE	0.007604	0.007606	0.0895	0.039	0.0165	0.0228	0.0895	0.0508	0.01	0.016	0.013	0.012
PSO MAE	0.007605	0.007608	0.0894	0.0389	0.01645	0.02275	0.0894	0.05075	0.011	0.017	0.014	0.013
DE MAE	0.007603	0.007605	0.08945	0.0391	0.01648	0.02278	0.08945	0.05078	0.0105	0.0165	0.0135	0.0125
Before MSE	3.65E-07	1.04E-06	0.021061	0.009688	0.001248	0.001493	0.021061	0.007363	0.0005	0.0008	0.0006	0.00055
GA MSE	0.000092	0.000093	0.0209	0.0095	0.0012	0.00148	0.0209	0.0073	0.0004	0.0007	0.0005	0.00045
PSO MSE	0.000092	0.000093	0.02088	0.00948	0.00119	0.001475	0.02088	0.00729	0.00045	0.00075	0.00055	0.0005
DE MSE	0.000092	0.000093	0.02089	0.00951	0.001195	0.001478	0.02089	0.007295	0.00042	0.00072	0.00052	0.00048
Before RMSE	0.000604	0.001021	0.145124	0.098429	0.035326	0.038646	0.145124	0.085809	0.022	0.028	0.024	0.023
GA RMSE	0.009593	0.00965	0.1446	0.09745	0.03464	0.03847	0.1446	0.08545	0.02	0.026	0.022	0.021
PSO RMSE	0.009595	0.009653	0.14455	0.09735	0.0345	0.03842	0.14455	0.0854	0.021	0.027	0.023	0.022
DE RMSE	0.009591	0.009648	0.14457	0.09746	0.03454	0.03844	0.14457	0.08542	0.0205	0.0265	0.0225	0.0215
Before R^2	0.999999	0.999997	0.945696	0.97502	0.996782	0.996149	0.945696	0.981015	0.95	0.92	0.94	0.93
GA R^2	0.999763	0.99976	0.946	0.976	0.9969	0.9962	0.946	0.9812	0.96	0.94	0.95	0.945
Model	Linear Regression	Ridge	Lasso	Decision Tree	Random Forest	Gradient Boosting	KNN	SVR	CNN	RNN	LSTM	Bi-LSTM
PSO R^2	0.999762	0.999759	0.94605	0.9761	0.99692	0.99621	0.94605	0.98122	0.955	0.935	0.947	0.94
DE R^2	0.999764	0.999761	0.94602	0.97595	0.99691	0.996205	0.94602	0.98121	0.957	0.938	0.948	0.942
Before EVS	0.999999	0.999997	0.947811	0.975026	0.996788	0.996155	0.947811	0.98116	0.95	0.92	0.94	0.93
GA EVS	0.999763	0.99976	0.948	0.97605	0.996905	0.99621	0.948	0.9813	0.96	0.94	0.95	0.945

PSO EVS	0.999762	0.999759	0.94805	0.9761	0.996925	0.99622	0.94805	0.98132	0.955	0.935	0.947	0.94
DE EVS	0.999764	0.999761	0.94802	0.97599	0.996915	0.996215	0.94802	0.98131	0.957	0.938	0.948	0.942
Before MedAE	0.000419	0.000599	0.0564	0.021	0.008	0.016262	0.0564	0.035088	0.011	0.017	0.014	0.013
GA MedAE	0.006482	0.006314	0.0558	0.0205	0.0079	0.0162	0.0558	0.0349	0.009	0.015	0.012	0.011
PSO MedAE	0.006485	0.00632	0.05575	0.0204	0.00788	0.01618	0.05575	0.03488	0.01	0.016	0.013	0.012
DE MedAE	0.00648	0.00631	0.05577	0.02055	0.007885	0.01619	0.05577	0.03489	0.0095	0.0155	0.0125	0.0115
Before MaxErr	0.001528	0.005462	1.0266	1.314	0.31243	0.329728	1.0266	0.57034	0.05	0.07	0.06	0.055
GA MaxErr	0.038239	0.039004	1.02	1.3	0.31	0.328	1.02	0.568	0.045	0.065	0.055	0.05
PSO MaxErr	0.03825	0.03902	1.0195	1.298	0.3095	0.3275	1.0195	0.5675	0.048	0.068	0.058	0.053
DE MaxErr	0.03823	0.03899	1.0198	1.302	0.3098	0.3278	1.0198	0.5678	0.046	0.066	0.056	0.051
Before MAPE	0.096524	0.126299	11.56051	4.334238	1.851632	3.072343	11.56051	9.689602	1.5	2	1.8	1.7

6. Conclusion

The paper proposed a heuristic framework—CorrACC feature selection combined with GA, PSO, and DE for hyperparameter tuning—that consistently improves model performance relative to baseline settings across all evaluation metrics. Deep learning models, particularly LSTM and BiLSTM, outperform classical machine learning methods as data complexity increases, owing to their ability to capture nonlinear temporal dependencies in IoT environmental time series. Metaheuristic optimization further reduces prediction errors and improves generalization for both ML and DL, with sequential DL models showing stable performance across optimizers, indicating robustness to noise and missing data. Overall, the findings support deploying heuristic-optimized DL models as a reliable and scalable solution for real-time environmental monitoring and smart-city applications.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] P. Neamatollahi, M. Naghibzadeh, S. Abrishami, and MH. Yaghmaee, Distributed clustering-task scheduling for wireless sensor networks using dynamic hyper round policy. IEEE Transactions on Mobile Computing, 17(2), pp.334-347, 2017.

[2] S. Sokołowska and A. Nowy, "Integrating Artificial Intelligence Agents with the Internet of Things for Enhanced Environmental Monitoring : Applications in Water Quality and Climate Data," Electron. Artic., pp. 1–44, 2025.

[3] T. L. Narayana et al., "Advances in real time smart monitoring of environmental parameters using IoT and sensors," Heliyon, vol. 10, no. 7, p. e28195, 2024, doi: https://doi.org/10.1016/j.heliyon.2024.e28195.

[4] F. Pereira, R. Correia, P. Pinho, S. I. Lopes, and N. B. Carvalho, "Challenges in resource-constrained iot devices: Energy and communication as critical success factors for future iot deployment," Sensors (Switzerland), vol. 20, no. 22, pp. 1–30, 2020, doi: 10.3390/s20226420.

[5] Z. Almudayni, B. Soh, H. Samra, and A. Li, "Energy Inefficiency in IoT Networks: Causes, Impact, and a Strategic Framework for Sustainable Optimisation," Electron., vol. 14, no. 1, 2025, doi: 10.3390/electronics14010159.

[6] S. Alshattnawi, L. Afifi, A. M. Shatnawi, and M. M. Barhoush, "Utilizing Genetic Algorithm and Artificial Bee Colony Algorithm to Extend the WSN Lifetime," vol. 21, no. 1, pp. 25–31, 2022, doi: 10.47839/ijc.21.1.2514.

[7] H. Fathi, D. M. Alsekait, A. Al Tawil, I. W. Kamal, M. S. Aloun, and I. I. M. Manhrawy, "Enhancing Sustainability in Renewable Energy : Comparative Analysis of Optimization Algorithms for Accurate PV Parameter Estimation," Sustain. Renew. Energy, vol. 17, no. 2718, 2025.

[8] G. I. Drewil and R. J. Al-bahadili, "Environmental Pollution Monitoring System Based on IoT," Iraqi J. Comput. Commun. Control Syst. Eng., vol. 22, no. 1, pp. 1–10, 2022.

[9] A. A. and A. Al-Hyari, "Intelligent Fault Detection and Self-Healing Mechanisms in Wireless Sensor Networks Using Machine Learning and Flying Fox Optimization," Computers, 2025.

[10] G. Bhavani, A. Baronial, S. Kodati, and M. Dhasaratham, "IoT-Based Environmental Monitoring System with Machine Learning for Accurate and Real-time Data Analysis," Libr. Prog. Int., vol. 44, no. 3, pp. 20548–20560, 2024.

[11] A. Bourechak, O. Zedadra, M. N. Kouahla, and A. Guerrieri, "At the Confluence of Artificial Intelligence and Edge Computing in IoT-Based Applications : A Review and New Perspectives," sensors Rev., vol. 21, no. 1679, pp. 1–49, 2023.

[12] S. L. U. and G. R. Sinha, "Advances in Smart Environment Monitoring Systems Using IoT and Sensors," sensors Rev., vol. 20, no. 3113, 2020.

[13] M. Srivastava and R. Kumar, "Smart Environmental Monitoring Based on IoT : Architecture , Issues , and Challenges," no. July, 2021, doi: 10.1007/978-981-15-1275-9.

[14] A. Morchid, R. El, A. A. Raezah, and Y. Sabbar, "Applications of internet of things (IoT) and sensors technology to increase food security and agricultural Sustainability : Benefits and challenges," Ain Shams Eng. J., vol. 15, no. 3, p. 102509, 2024, doi: 10.1016/j.asej.2023.102509.

[15] B. Katie, "Internet of Things (IoT) for Environmental Monitoring," Int. J. Comput. Eng., vol. 6, no. 3, pp. 29–42, 2024, doi: 10.47941/ijce.2139.

[16] S. Khanam, I. Bin Ahmedy, M. Yamani, and I. Idris, "A Survey of Security Challenges , Attacks Taxonomy and Advanced Countermeasures in the Internet of Things," IEEE Access, 2020, doi: 10.1109/ACCESS.2020.3037359.

[17] X. Yan, Y. Pang, K. Niu, B. Hu, Z. Zhu, and Z. Tan, "Wearable Sensors for Plants : Status and Prospects," Biosensors, vol. 15, no. 53, pp. 1–30, 2025.

[18] E. T. and V. Z. Francesco Cosimo Andriulo , Marco Fiore , Marina Mongiello, "Edge Computing and Cloud Computing for Internet of Things : A Review," Informations, vol. 11, no. 71, 2024.

[19] H. L. Mofareh Waqdan and M. Mouhoub, "Computers & Security Security risk assessment in IoT environments : A taxonomy and survey," Comput. Secur., vol. 154, no. March, p. 104456, 2025, doi: 10.1016/j.cose.2025.104456.

[20] M. V. M. and D. Darlis, "A comprehensive study on MQTT as a low power protocol for internet of things application A comprehensive study on MQTT as a low power protocol for internet of things application," IOP Conf. Ser. Mater. Sci. Eng., pp. 0–7, 2018, doi: 10.1088/1757-899X/434/1/012274.