Wasit Journal for Pure Science

Journal Homepage: https://wjps.uowasit.edu.iq/index.php/wjps/index
e-ISSN: 2790-5241 p-ISSN: 2790-5233

Experimental Analysis of Random Forest and LSTM Models for Optimizing CPU Efficiency in Cloud Computing

Sajjad A. Ajeel[1]*, Sundos A. Hameed[2]

[1,2]Department of Computer Science, Al-Mustansiriyah University, IRAQ

*Corresponding Author: Sajjad A. Ajeel

DOI: https://doi.org/10.31185/wjps.893
Received 11 July 2025; Accepted 21 September 2025; Available online 30 September 2025

ABSTRACT: Cloud computing offers scalable, on-demand computational resources; however, efficient CPU utilization remains a critical challenge, as insufficient provisioning can lead to increased latency and higher energy consumption. This study aims to evaluate and optimize CPU performance in cloud systems using two machine learning models—Random Forest (RF) and Long Short-Term Memory (LSTM)—combined with the Multi-Objective Grey Wolf Optimizer (MOGWO) for hyperparameter tuning in time-series prediction. The models were assessed on the Google 2019 cluster trace using standard evaluation metrics, including Accuracy, Precision, Recall, and F1-score. Experimental results demonstrate that MOGWO-based optimization significantly improves model performance. The accuracy of the RF model increased from 65.15% to 89.70%, while the LSTM model improved from 95.02% to 98.60%, with corresponding gains in precision, recall, and F1-score. These findings confirm the effectiveness of the optimized models in predicting CPU usage patterns and enhancing CPU resource management in cloud computing environments.

Keywords: Cloud computing, CPU efficiency, Random Forest, LSTM, machine learning, resource optimization, performance prediction, time-series analysis, Metaheuristic Optimization, Cloud Resource Scheduling.

1. INTRODUCTION

Modern information technology relies heavily on cloud computing, which enables on-demand access to computational resources, including storage and processing capacity, within a virtualized environment. As cloud-based systems continue to expand, the need for effective resource management has become critical, particularly in optimizing CPU usage to enhance data center performance and reduce energy costs [1].

CPU resource management in cloud environments is increasingly complex due to the natural variability of workload patterns and evolving service demands. Fixed provisioning techniques have proven inefficient, often resulting in either underutilization or performance degradation. Consequently, dynamic resource allocation strategies have emerged, offering near-instant provisioning based on workload trends and system feedback [2].

Dynamic approaches supported by machine learning (ML) algorithms, such as Random Forests (RF), leverage historical workload data to predict future resource requirements. These models are increasingly integrated with scheduling methods to optimize CPU utilization and prevent Service Level Agreement (SLA) violations [3].

Beyond conventional ML models, neural networks such as Long Short-Term Memory (LSTM) networks have demonstrated exceptional capability in capturing nonlinear, temporal dependencies within workload data. By effectively modeling sequential patterns, LSTM networks provide more accurate CPU demand forecasts, enabling improved decision-making in task allocation and scheduling [4].

Algorithmic optimization has become indispensable for enhancing resource allocation, as it identifies optimal system parameter values. Recent advancements in reinforcement learning and multi-objective optimization have facilitated strategies that balance multiple goals, including runtime reduction, energy efficiency, and load distribution among virtual machines [5].

Energy efficiency is a critical factor in CPU optimization. Techniques such as virtual machine (VM) consolidation, workload clustering, and predictive migration allow data centers to minimize the number of active physical machines, reducing energy consumption and promoting sustainability [6].

Recent studies highlight that integrating clustering, machine learning, and optimization can significantly improve both computational and energy efficiency. Notably, models that combine supervised learning for prediction with metaheuristic optimization for control have achieved strong results in maintaining balanced cloud operations [7].

As cloud environments grow increasingly complex and large-scale, there is a pressing demand for intelligent, adaptive, multi-objective frameworks that integrate ML models with metaheuristic optimization techniques [8].

While standalone models provide valuable insights, hyperparameter optimization can substantially enhance their performance. In this study, the Multi-Objective Grey Wolf Optimizer (MOGWO) is employed to optimize the hyperparameters of RF and LSTM models. MOGWO, inspired by the social hierarchy and hunting strategies of grey wolves, is particularly effective for multi-objective problems requiring trade-offs between competing performance measures [9].

This research makes two primary contributions. First, it systematically integrates MOGWO with RF and LSTM models in a multi-objective environment, enabling optimal CPU efficiency prediction in cloud computing. Second, it conducts a comparative evaluation of the optimized models using the Google Cluster 2019 trace, a large-scale, realistic dataset that captures workload dynamics. Together, these contributions—methodological integration and empirical benchmarking—demonstrate both the theoretical significance and the practical relevance of this work for cloud resource management.

2. RELATED WORK

Ahamed et al. [2023] [10] conducted a comparative technical analysis of deep learning (DL) architectures—including LSTM, RNN, CNN, and MLP—for cloud workload prediction. Their findings indicated that LSTM achieved the best performance on time-series data, producing the lowest RMSE values and the highest prediction accuracy. They also emphasized the importance of clean and consistent workload data to enable reliable and repeatable model training.

Sabzipour et al. [2023] [11] compared the effectiveness of an LSTM neural network with a traditional hydrological model in predicting streamflow for a Canadian catchment. Without data assimilation, LSTM demonstrated significant improvements in accuracy, reducing MAE to 25 m^3/s—a major advancement, particularly in short-term predictions. Moreover, it streamlined computations and lowered CPU consumption, highlighting its potential for effective prediction in dynamic environments.

Jeyaraman et al. [2024] [12] evaluated the efficiency of LSTM in combination with Monte Carlo Tree Search for SLA compliance in high-traffic network scenarios. Their results showed that LSTM improved accuracy by 10–15% and reduced error rates by approximately 9.5–10.2%, particularly under heavy traffic loads. This demonstrated the model's capability to enhance overall network utilization while minimizing CPU and memory consumption.

Gong et al. [2024] [13] proposed an ML-based approach for dynamic resource allocation and VM migration optimization. By integrating DQN for real-time migration decisions with predictive models such as LSTM and GRU for CPU and memory usage, their framework enhanced load balancing, improved resource utilization, and reduced energy consumption. This study underscored the central role of ML in intelligent cloud resource management.

Theodore [2024] [14] focused on ROI optimization in cloud environments through resource cost management strategies. The adoption of right-sizing, auto-scaling, and FinOps models yielded cost savings of up to 30% without compromising performance. The study also highlighted that AI-enabled automation and cost-optimization tools enhanced visibility, governance, and financial accountability in cloud operations.

Amarnath [2025] [15] examined contemporary resource allocation frameworks incorporating intelligent scheduling, auto-scaling, and ML-based decision-making. The study confirmed that combining metaheuristic algorithms with ML models significantly reduced resource usage and SLA violations. Predictive scaling and anomaly detection were emphasized as critical techniques for maintaining CPU efficiency under increasing workloads.

Wang and Yang [2025] [16] developed an intelligent resource allocation mechanism integrating deep learning (LSTM) with deep reinforcement learning (DQN) in an end-to-end framework. Their approach achieved substantial improvements in production cloud environments, increasing resource utilization by 32.5%, reducing response time by 43.3%, and cutting operational costs by 26.6%.

Although these studies present valuable insights and practical approaches to resource management, a gap remains in the form of a comprehensive comparative analysis of ML models—particularly with respect to CPU usage optimization in cloud systems. The present research addresses this gap by evaluating and benchmarking supervised learning models such as Decision Trees, Random Forests, and Neural Networks for workload pattern prediction and CPU efficiency improvement across diverse cloud scenarios.

3. PROPOSED METHOD

In this research, a two-step framework is developed to assess and optimize CPU efficiency in cloud computing systems using machine learning and optimization techniques. In the first step, two predictive models—Random Forest (RF) and Long Short-Term Memory (LSTM)—are trained and evaluated according to standard best practices. In the second step, the Multi-Objective Grey Wolf Optimizer (MOGWO) is employed to fine-tune the hyperparameters of both models, thereby enhancing prediction accuracy, improving generalization, and increasing resource efficiency. The overall architecture of the proposed system is illustrated in Figure 1:

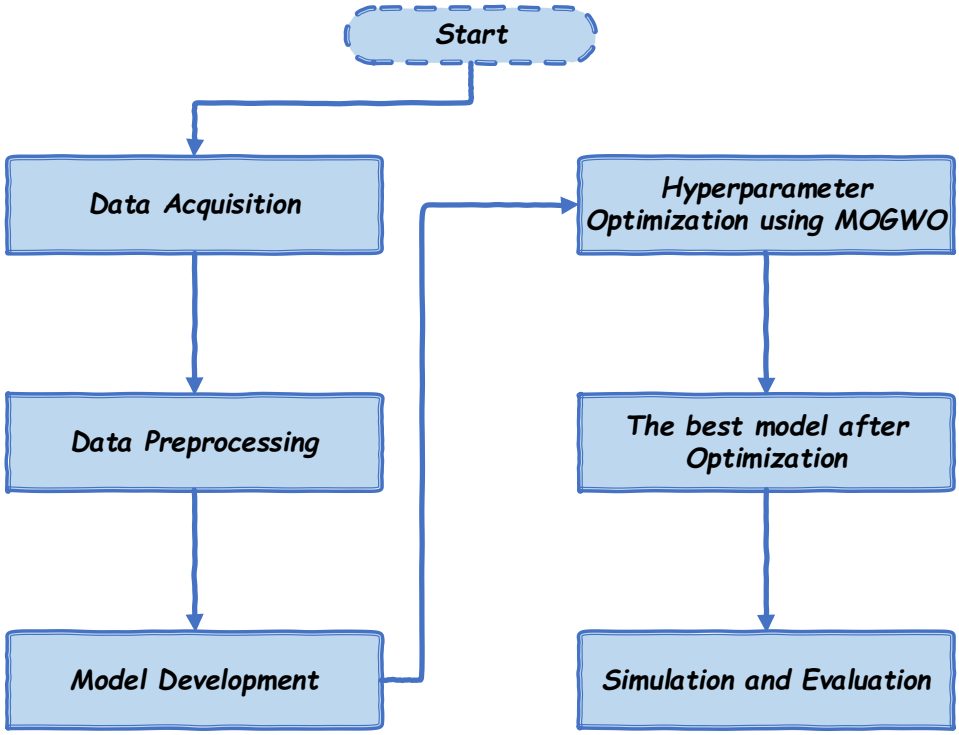

FIGURE 1. - The structure of the proposed system after integrating MOGWO optimization.

3.1 DATASET

The dataset used in this study is the Google Cluster 2019 sample, which contains 2.4 TiB of workload traces collected from eight clusters worldwide. The trace includes comprehensive information on submitted jobs, scheduling events, and resource usage data for tasks executed within these clusters. Borg, the cluster management system, supports two types of resource requests: jobs (collections of one or more tasks that define the computations a user wishes to execute) and allocation sets (collections of one or more allocations or allocation instances that represent reserved resources for executing those computations) [17].

3.2 PREPROCESSING

Preprocessing is a critical phase to ensure that the data is of high quality, consistent, and suitable for machine learning tasks. In this study, a systematic preprocessing pipeline was employed, consisting of three main steps: (i) handling missing and corrupted values, (ii) feature selection and engineering, and (iii) data normalization and scaling. This procedure ensured that the dataset was free of inconsistencies, contained only the most relevant features, and was appropriately scaled for use with both traditional ML models and neural networks. The details of each stage are outlined in the following subsections:

3.2.1 Handling Missing Values

Data cleaning is a crucial preprocessing step, as the quality of input data directly influences the performance of machine learning models in both training and forecasting. To develop robust and reliable models, it is essential to

ensure that the dataset is free from conflicts, missing entries, and infinite values. In this study, two complementary approaches were applied to handle missing values:

Deletion: Rows containing infinite values (e.g., inf, -inf) or severely corrupted records were removed, as such entries could introduce instability during model training.

Imputation: For partially missing values (e.g., NaN in non-critical fields), statistical imputation techniques were employed. Missing entries were replaced using representative measures such as the mean or median, ensuring minimal distortion of the dataset.

This two-step approach preserved valuable information, maintained data integrity, and provided a complete and valid dataset for use in both training and evaluation of the machine learning models.

3.2.2 Data Normalization

Normalization is a fundamental preprocessing step aimed at rescaling input features to a consistent range, ensuring that machine learning models—particularly neural networks, which are sensitive to feature magnitude—perform effectively. In this study, all input variables were normalized to promote proportional feature contribution and accelerate model convergence.

Two commonly used normalization techniques were applied:

Min-Max Scaling: Transforms values to a range between 0 and 1 while preserving the relative order of the data.

Standardization: Centers the data around the mean and scales it by the unit variance, making it particularly suitable for features following a Gaussian distribution.

Normalization is essential for successful model training, as it ensures that all features operate on the same scale. This not only speeds up learning and stabilizes training by mitigating issues such as vanishing or exploding gradients but also enhances model generalization to unseen data. Moreover, normalization keeps the inputs of activation functions within appropriate ranges, thereby improving gradient flow and amplifying the effectiveness of regularization techniques such as dropout and batch normalization.

3.2.3 Feature Selection

Feature selection is a critical component of the preprocessing phase, aimed at improving model accuracy while reducing computational complexity. By removing irrelevant or redundant inputs and retaining only the most informative variables, this process minimizes noise and enhances the dataset's predictive value. In this study, feature selection was informed by domain expertise, systematic feature engineering, and empirical testing.

In addition to the original features, new variables were derived by transforming distribution-based columns and summarizing them through statistical descriptors to highlight key patterns. These engineered features captured important temporal and performance-related characteristics directly influencing CPU scheduling and utilization in the cloud, thereby improving the ability to predict system performance.

Guided by domain knowledge and preliminary statistical analysis, only the most relevant attributes were retained for model training. The final set of features demonstrated strong correlations with CPU activity, which not only enhanced the predictive accuracy of the models but also simplified training and evaluation.

3.3 MODELS DEVELOPMENT

Deep learning and machine learning models were employed to forecast task duration and CPU utilization in cloud architectures. The models selected for evaluation included Decision Tree (DT), Random Forest (RF), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN).

For training and evaluation, the dataset was partitioned using the hold-out validation approach, with 70% allocated for training and 30% for testing. This random split ensured a balanced assessment of model generalization while preserving sufficient data for effective learning. The 70/30 ratio, widely adopted in machine learning applications, provides an optimal trade-off between training sufficiency and testing reliability.

During the training phase, models were fitted and their internal parameters optimized using the training set. Performance was then evaluated on the testing set to measure their ability to handle previously unseen data. The overall framework for CPU efficiency prediction is illustrated in Figure 2.

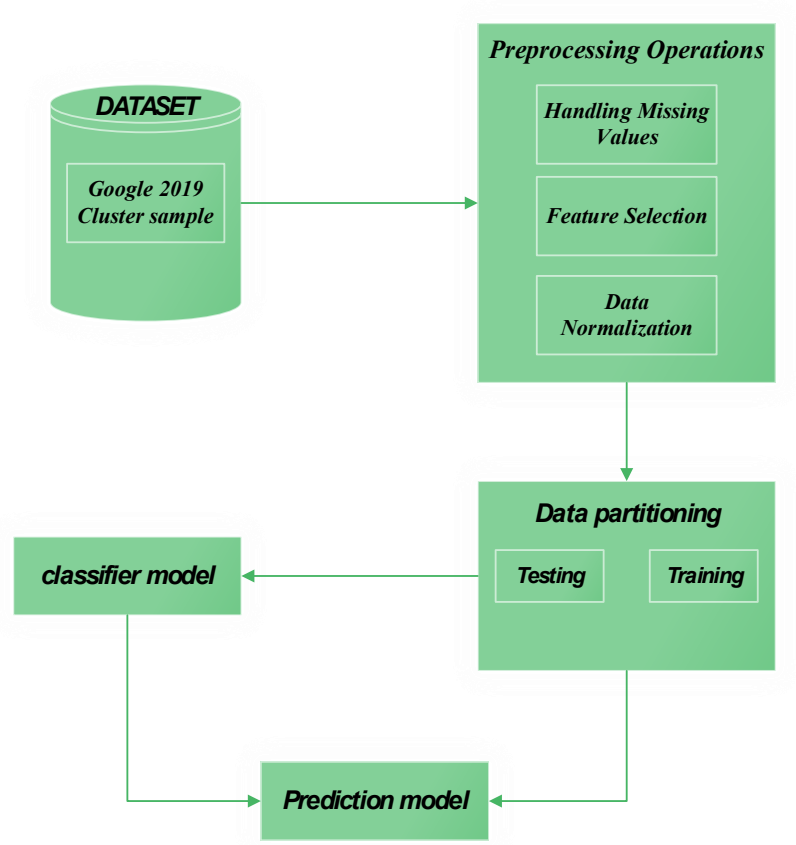

FIGURE 2. - Model structure of CPU Efficiency Prediction

3.4 RANDOM FOREST MODEL WITH MOGWO OPTIMIZATION

The Random Forest (RF) algorithm was first employed as a baseline model for predicting CPU efficiency using multiple system-level features. As an ensemble method, RF constructs a collection of decision trees and aggregates their predictions, thereby improving accuracy and reducing the risk of overfitting. While the default parameters provided a useful benchmark for evaluating performance, further improvements were expected through hyperparameter tuning.

To enhance the predictive capability of RF, hyperparameters were optimized using the Multi-Objective Grey Wolf Optimizer (MOGWO). Key parameters tuned included the number of estimators (trees), the maximum tree depth, and the minimum number of samples required at leaf nodes. MOGWO guided the optimization process through a multi-objective fitness function that simultaneously maximized classification accuracy and F1-score while minimizing validation loss.

In this framework, each solution (wolf) in the MOGWO population represented a unique configuration of RF hyperparameters. Candidate solutions were evaluated using cross-validation on the training set, and non-dominated solutions were stored in an external archive to ensure they were not superseded by others. Upon convergence, the optimal parameter set was selected from the Pareto front and used to retrain the RF model, thereby improving its classification performance and efficiency in CPU usage prediction. The main stages of the RF workflow are outlined in Algorithm (1).

Algorithm (1): The main stages of the RF.
Input: Preprocessed dataset (features and labels) MOGWO-optimized hyperparameters
Output: Predicted class labels for the test set

Begin

Step 1: Load and preprocess the dataset

Step 2: Split the dataset into training and testing sets

Step 3: Use MOGWO to optimize the hyperparameters of the RF model

3.1: Define multi-objective function (maximize accuracy, F1-score; minimize validation loss)

3.2: Initialize wolf population and iterate to explore search space

3.3: Update Pareto front with non-dominated solutions

Step 4: Train the Random Forest model using the best hyperparameters found by MOGWO

Step 5: Predict the target labels for the test dataset

Step 6: Evaluate the model using the following metrics:

Accuracy, Precision, Recall, F1 Score

Step 7: Save the optimized model and evaluation results

End

3.5 LSTM MODEL WITH MOGWO OPTIMIZATION

To capture sequential dependencies in CPU workload data, a neural network based on Long Short-Term Memory (LSTM) was developed and optimized using the Multi-Objective Grey Wolf Optimizer (MOGWO). The input layer receives 14 features, which are reshaped into a time-series format suitable for LSTM processing. The architecture consists of a single LSTM layer, followed by batch normalization, two fully connected dense layers, and a final sigmoid layer for binary classification.

MOGWO was employed to optimize four key hyperparameters: the number of units in the LSTM layer, the number of neurons in each dense layer, and the learning rate. The optimization process used a composite objective function comprising three conflicting objectives: minimizing validation loss while maximizing validation accuracy and F1-score.

During each iteration, the LSTM model was trained with the candidate hyperparameter settings, and solutions (wolves) were evaluated based on performance metrics computed on the validation set. The most promising non-dominated solutions were stored in an external archive, forming a Pareto front. Upon convergence, the optimal hyperparameter set was selected, and the final LSTM model was retrained to achieve enhanced predictive performance. The main stages of LSTM optimization using MOGWO are outlined in Algorithm (2).

Algorithm (2): LSTM Optimization Using MOGWO

Input:

Preprocessed dataset (features and labels)

Search space: ranges for LSTM and Dense units, learning rate

Objective weights: α,β,γ

Output:

Optimized LSTM model

Begin

Step 1: Define Objective Function.

Step 2: Define Search Space.

Step 2.1: Set the ranges for hyperparameters (lstm_units, Dense layers, learning rate).

Step 2.2: Define a fitness function combining validation loss, accuracy, and F1 score.

Step 3: Initialize Wolves Population

Step 3.1: Randomly generate N wolves, each representing a set of hyperparameters within the defined bounds.

Step 4: Initialize Pareto Archive

Create an empty list to store non-dominated solutions.

Step 5: Start Optimization Loop (for max_iter iterations)

For each iteration:

a. Evaluate all wolves using the objective function

b. Update Pareto Archive with non-dominated wolves

c. Identify Alpha, Beta, Delta wolves from archive

d. Update each wolf's position based on Grey Wolf rules (A, B, D)

e. Clip all updated parameters within valid search bounds

Step 6: Select Best Solution

Choose the solution with the lowest fitness score from the archive

Step 7: Rebuild Final LSTM Model

Use the best parameters to retrain the LSTM on the full training set

Step 8: Evaluate Final Model

Compute and report:

Accuracy, Precision, Recall, F1-score

End

3.6 EVALUATION METRICS

Four standard classification evaluation metrics were used to assess the performance of the proposed models before and after optimization: Accuracy, Precision, Recall, and F1-score.

- Accuracy measures the overall correctness of the model by calculating the ratio of correctly predicted instances to the total number of instances.
- Precision evaluates the proportion of correctly predicted positive cases among all instances predicted as positive, reflecting the reliability of positive predictions.
- Recall (sensitivity) indicates the model's ability to correctly identify actual positive cases.
- F1-score is the harmonic mean of precision and recall, making it particularly useful for evaluating performance on imbalanced datasets.

Together, these metrics provide a comprehensive assessment of classification quality and allow for a systematic comparison of all models, including both baseline and MOGWO-optimized versions [18, 19].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F1\ score = \frac{Precision\ x\ Recall}{Precision + Recall}\ x2 \quad (4)$$

The values in TP represent correctly classified positive samples while TN indicates correctly classified negative samples and both FP and FN account for the samples misclassified as positive or negative respectively.

4 EXPERIMENTAL RESULTS AND DISCUSSION

This section presents and analyzes the experimental results of applying RF and LSTM models—both before and after MOGWO optimization—on the cloud workload dataset. Model performance is evaluated using standard classification metrics to demonstrate improvements and their implications for CPU efficiency prediction.

The experiments were conducted on a personal computer running Windows 10, equipped with an Intel i7 processor and 16 GB of DDR4 RAM. Implementation was performed using Python 3.12 within Jupyter Notebook, and experiments were also conducted on the Kaggle cloud platform to ensure reproducibility and facilitate experimentation.

4.1 BASELINE PERFORMANCE OF RF AND LSTM (BEFORE OPTIMIZATION)

The Random Forest (RF) and Long Short-Term Memory (LSTM) models were compared before optimization using four standard classification metrics: Accuracy, Precision, Recall, and F1-score. With default hyperparameters, the RF model achieved an accuracy of 65.15%, which, along with the other metrics, indicated limited generalizability and an inability to capture the specific patterns in the dataset. In contrast, the LSTM model performed significantly better, achieving an accuracy of 95.02% and strong scores across precision, recall, and F1-score. These baseline results demonstrate LSTM's superior ability to capture temporal CPU workload patterns while highlighting the potential benefits of further hyperparameter optimization for both models.

FIGURE 3. Training and validation performance metrics of the LSTM model: (a) Training and Validation Loss, (b) Training and Validation Accuracy.

4.2 PERFORMANCE AFTER MOGWO OPTIMIZATION

Using the Multi-Objective Grey Wolf Optimizer (MOGWO), both the Random Forest (RF) and Long Short-Term Memory (LSTM) models showed substantial improvements across all evaluation metrics. The optimization process simultaneously targeted multiple objectives—Accuracy, Precision, Recall, and F1-score—by tuning several key hyperparameters.

For the RF model, optimization resulted in a notable performance increase, with accuracy rising to 89.70%, accompanied by significant improvements in the other evaluation metrics. This demonstrates that MOGWO effectively identifies a balanced and efficient set of hyperparameters that better capture the underlying structure of the data.

Similarly, the LSTM model achieved exceptional results, with accuracy increasing to 98.60% after optimization. The model also showed strong precision, recall, and F1-score, indicating robust generalization and accurate prediction

on unseen data. These results underscore the effectiveness of MOGWO in enhancing model behavior and maximizing CPU prediction efficiency in cloud environments.

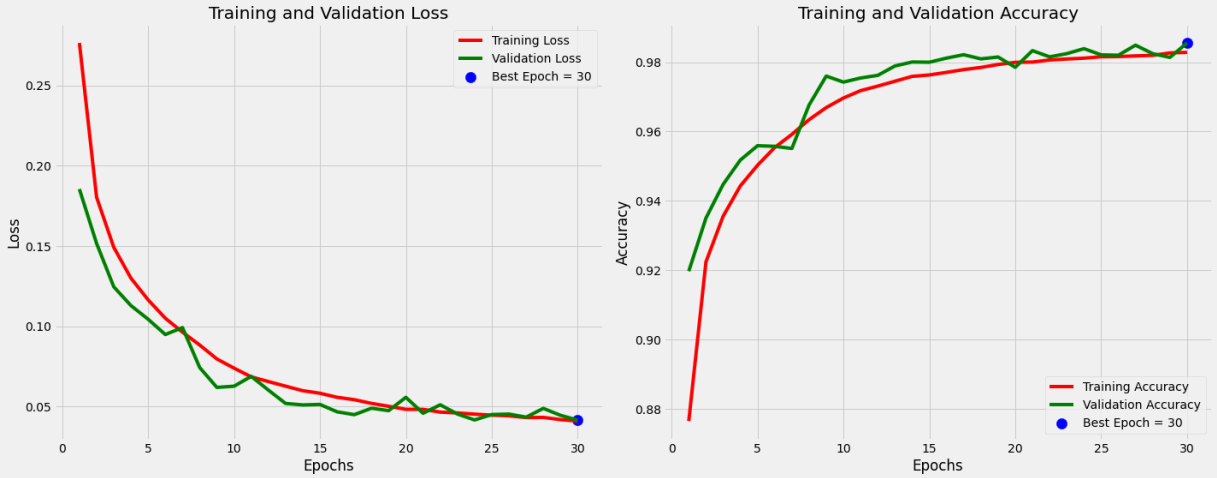

FIGURE 4. Training and validation performance metrics of the LSTM model after optimization: (a) Training and Validation Loss, (b) Training and Validation Accuracy.

Table 1. - Model performance comparison

Algorithm	Accuracy	Precision	Recall	F1- Score
RF	0.6515	0.2339	0.2285	0.2312
LSTM	0.9502	0.9116	0.8668	0.8886
RF+MOGWO	0.8970	0.9476	0.5830	0.7219
LSTM+MOGWO	0.9856	0.9565	0.9817	0.9689

Table 1 presents a comparative summary of the performance of Random Forest (RF) and Long Short-Term Memory (LSTM) models before and after optimization using the Multi-Objective Grey Wolf Optimizer (MOGWO). The baseline RF model exhibited limited performance, with an accuracy of 65.15% and a low F1-score, while the baseline LSTM achieved substantially higher results (95.02% accuracy, 0.8886 F1-score). After optimization, RF improved to 89.70% accuracy, and LSTM reached 98.56% accuracy with an F1-score of 0.9689.

These results demonstrate that MOGWO significantly enhances the predictive performance of both models, with LSTM remaining the best-performing model for CPU efficiency prediction. Overall, the findings confirm that although LSTM is inherently superior for time-series CPU prediction, MOGWO-based hyperparameter optimization substantially improves accuracy, generalization, and computational efficiency, making both models more practical for cloud computing applications.

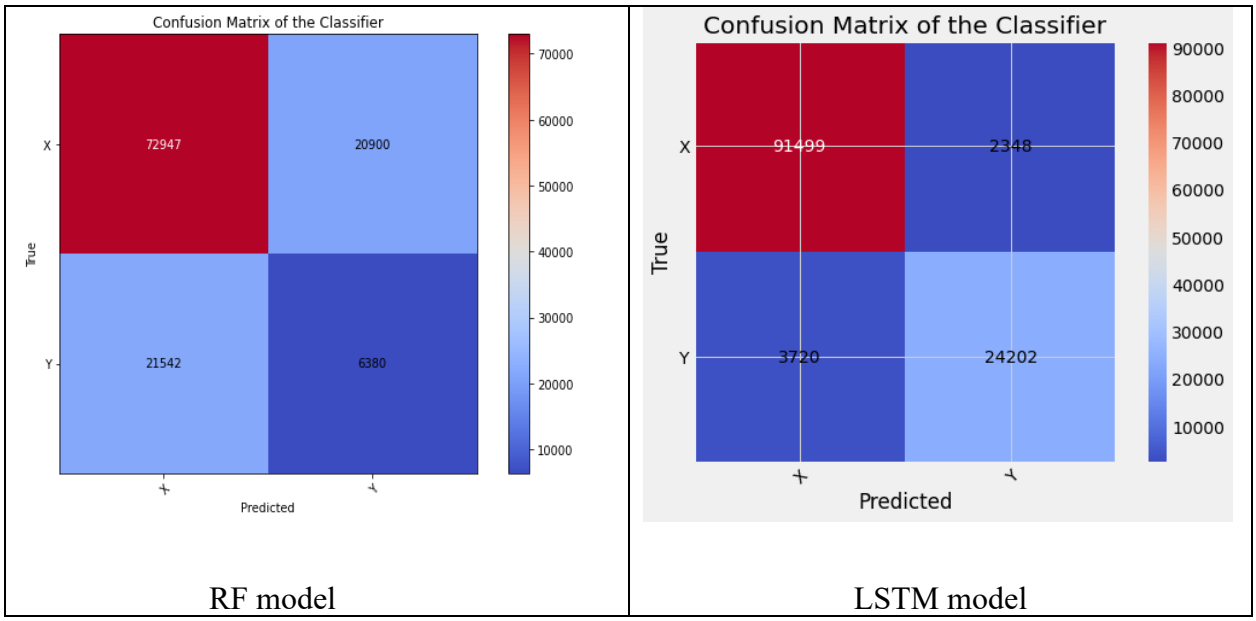

FIGURE 5. - Confusion Matrices for RF and LSTM Models Before MOGWO Optimization

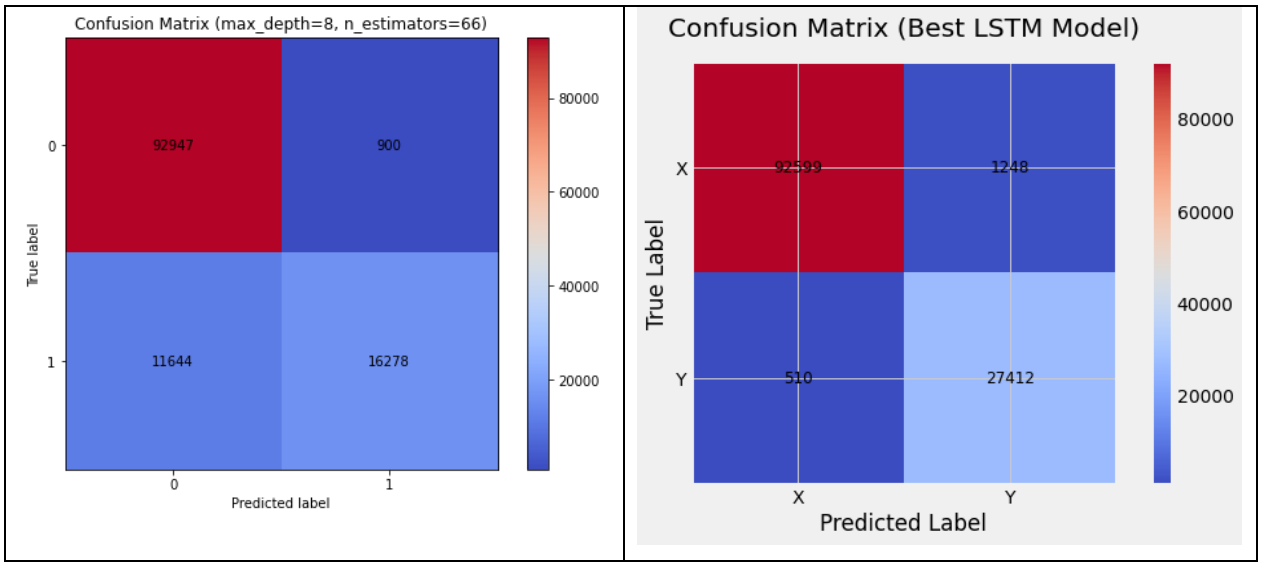

FIGURE 6. - Confusion Matrices for RF and LSTM Models After MOGWO Optimization

4.3 EFFECT OF OPTIMIZATION ON CPU EFFICIENCY

The integration of MOGWO-based optimization substantially enhanced the system's ability to evaluate and predict CPU efficiency within the cloud computing environment. The optimized framework enabled more accurate identification of task behaviors, resource requirements, and potential inefficiencies, reflecting the increased predictive accuracy of both RF and LSTM models. Consequently, the system became more effective in supporting intelligent decision-making related to CPU scheduling and load balancing.

Improved classification and forecasting accuracy reduced the risk of resource misallocation or waste, thereby enhancing system responsiveness and energy efficiency. These results demonstrate that applying a multi-objective

optimization approach not only improves model performance metrics but also delivers tangible operational benefits, enhancing the overall efficiency and reliability of cloud computing infrastructure.

5 CONCLUSION AND FUTURE WORK

The present study demonstrates that combining machine learning and deep learning models with the Multi-Objective Grey Wolf Optimizer (MOGWO) is an effective strategy for achieving higher accuracy in predicting CPU efficiency in cloud computing environments. Experimental results show that MOGWO-based hyperparameter tuning significantly enhanced the performance of both models across key evaluation metrics. The accuracy of the RF model increased from 65.15% to 89.70%, a gain of 24.55 percentage points, while the LSTM model improved from 95.02% to 98.60%, a 3.58-point increase, with consistent gains in precision, recall, and F1-score. These improvements translated into more reliable classification and better-informed decisions regarding CPU resource allocation.

Despite these positive outcomes, several limitations exist. The optimization process was computationally intensive, particularly for the LSTM model, due to extensive training cycles. Additionally, the employed dataset may not comprehensively represent all types of cloud workloads, which could limit the generalizability of the findings. Moreover, only a subset of hyperparameters was optimized, leaving other architectural parameters—such as dropout rates and the number of layers—unexplored.

Future work will focus on expanding the dataset to include real-time cloud traces and diverse workload patterns from production-scale sources, such as Google Cloud Traces or Microsoft Azure datasets. Dynamic hyperparameter adaptation techniques, including surrogate modeling or early stopping, will be explored to reduce computational costs. Architectural tuning using Neural Architecture Search (NAS) will also be investigated to optimize the number of LSTM layers, dropout rates, and activation functions. Finally, the proposed models will be implemented in a cloud-simulated environment to evaluate their performance under real-time CPU demand fluctuations, time constraints, and prioritization policies.

REFERENCES

1. S. S. Panwar, M. M. S. Rauthan, V. Barthwal, N. Mehra, and A. Semwal, “Machine learning approaches for efficient energy utilization in cloud data centers,” Procedia Computer Science, vol. 235, pp. 1782–1792, 2024.
2. L. Li and X. Gao, “Profit-Efficient Elastic Allocation of Cloud Resources Using Two-Stage Adaptive Workload Prediction,” *Applied Sciences*, vol. 15, no. 5, 2025.
3. R. Khan, “Dynamic load balancing in cloud computing: Optimized RL-based clustering with multi-objective optimized task scheduling,” Processes, vol. 12, no. 519, 2024.
4. R. Khan, "Dynamic Load Balancing in Cloud Computing: Optimized RL-Based Clustering with Multi-Objective Optimized Task Scheduling," *Processes*, vol. 12, no. 519, 2024.
5. S. A. M. Mostafa, A. Mohan, and J. Wang, “SLA-MORL: SLA-Aware Multi-Objective Reinforcement Learning for HPC Resource Optimization,” *arXiv preprint arXiv:2508.03509*, 2025.
6. X. Liu, J. Wu, and S. Liu, “A prediction-based multi-objective VM consolidation approach for cloud data centers,” *SSRN*, Tech. Rep. 4678941, 2024. [Online]. Available: https://www.ssrn.com/abstract=4678941.
7. Y. Wang and X. Yang, “Intelligent resource allocation optimization for cloud computing via machine learning,” *arXiv preprint arXiv:2504.03682*, 2025.
8. V. Dakić, M. Kovač, and J. Slovinac, “Evolving high-performance computing data centers with Kubernetes, performance analysis, and dynamic workload placement based on machine learning scheduling,” Electronics, vol. 13, no. 2651, 2024.
9. Y. Gong, R. A. Adjei, G. Tao, Y. Zeng, and C. Fan, “An improved multi-objective Grey Wolf Optimizer for aerodynamic optimization of axial cooling fans,” Applied Sciences, vol. 15, no. 9, p. 5197, May 2025. [Online]. Available: https://doi.org/10.3390/app15095197
10. Z. Ahamed, M. Khemakhem, F. Eassa, F. Alsolami, and A. S. A.-M. Al-Ghamdi, “Technical study of deep learning in cloud computing for accurate workload prediction,” Electronics, vol. 12, no. 3, p. 650, 2023.
11. B. Sabzipour, et al., “Comparing a long short-term memory (LSTM) neural network with a physically-based hydrological model for streamflow forecasting over a Canadian catchment,” Journal of Hydrology, vol. 627, Article 130380, 2023.
12. J. Jeyaraman, S. V. Bayani, and J. N. Arasu Malaiyappan, “Optimizing resource allocation in cloud computing using machine learning,” European Journal of Technology, vol. 8, no. 3, pp. 12–22, 2024.
13. Y. Gong, et al., “Dynamic resource allocation for virtual machine migration optimization using machine learning,” in Conf. on Cloud Computing Technologies, 2024.

14. M. Theodore, “Strategies for efficient cloud resource management and cost optimization,” Journal of Engineering Science, vol. 1, no. 1, pp. 186–196, 2024.
15. V. K. Amarnath, “Engaging techniques for effective resource allocation in cloud computing: Improving performance, availability, and cost management,” International Journal on Science and Technology (IJSAT), vol. 16, no. 1, pp. 1–8, 2025.
16. Y. Wang and X. Yang, “Intelligent resource allocation optimization for cloud computing via machine learning,” Advances in Computer, Signals and Systems, vol. 9, no. 1, pp. 55–60, 2025.
17. F. H. Bappy, T. Islam, T. S. Zaman, R. Hasan, and C. Caicedo, “A deep dive into the Google cluster workload traces: Analyzing the application failure characteristics and user behaviors,” in Proc. 2023 10th Int. Conf. Future Internet of Things and Cloud (FiCloud), Aug. 2023, pp. 103–108.
18. E. Hato, Z. S. Abduljabbar, and Z. J. Ahmed, “Comparative analysis for bag of features (BoF) performance,” Iraqi Journal of Science, vol. 65, pp. 4606–4622, 2024.
19. P. Panda, S. K. Bisoy, S. Kautish, R. Ahmad, A. Irshad, and N. Sarwar, “Ensemble Classification Model With CFS-IGWO–Based Feature Selection for Cancer Detection Using Microarray Data,” *International Journal of Telemedicine and Applications*, vol. 2024, no. 1, Art. no. 4105224, 2024.