Journal of Global Research in Electronics and Communication
Volume 2, No. 06, June 2026
Available Online at: www.jgrec.info
RESEARCHPAPER

Minimum Response Time Optimization in Cloud-Based IoT Systems

Hitesh Kumar Sharma
Research Scholar
Amity university, Gwalior
Hiteshkumar1706@gmail.com

Dr Samta Jain Goyal
Associate professor
Amity university
Madhya Pradesh, Gwalior
sjgoyal@gwa.amity.edu

Dr. Sumit Kumar
Associate Professor
G N S University Sasaram, Bihar
sumit170787@gmail.com

***Abstract*—Cloud computing and IoT technologies are evolving quickly, leading to a growing need for intelligent systems that can process the dynamic workloads with minimal latency. However, effective response time and resource utilization are still huge challenges in cloud-IoT environments. This study suggests a minimum response time optimization framework for cloud-based IoT systems to achieve latency-aware resource scheduling and adaptive optimization under dynamic workloads. Data preprocessing, feature engineering, exploratory data analysis, and normalization follow the utilization of the Multi-Tier IoT Resource Allocation Dataset from Kaggle. The metrics MAE, MSE, RMSE, and R^2 are used for training and evaluation of Random Forest (RF), Gradient Boosting (GB), Stochastic Gradient Descent (SGD), and Deep Neural Network (DNN). Experimental outcomes show that the DNN model outperforms the base models including RF, GB, and SGD with an MAE of 5.9605 ms and an R^2 score of 0.9830. Results show that the suggested strategy is more efficient and performs better overall in terms of response time minimization, latency optimization, throughput improvement, energy efficiency, and SLA-aware QoS enhancement. The novelty of this work is in combining the three elements of response time prediction, adaptive resource allocation, and optimization analysis in a single framework, which can effectively enhance the performance and reliability of cloud-based IoT systems.**

Keywords*—Internet of Things (IoT), Edge-Cloud computing, Response Time, QoS Improvement, Latency Reduction, Resource Allocation.

I. INTRODUCTION

The IoT has become so pervasive that billions of devices are deployed that are constantly generating vast quantities of different data. Intelligent transportation, smart cities, healthcare systems, and industrial automation are all examples of applications that require efficient data processing and real-time decision-making [1][2]. Existing computing systems face substantial challenges in having to process a large number of IoT devices with respect to latency, resource allocation, scalability and optimization in response time [3][4]. To tackle these challenges, cloud computing has proven to be a suitable platform by offering centralized data management, computational resources, and scalable storage in IoT settings [5][6]. However, as workloads increase in complexity and resources are requested on the fly, responses may be delayed, affecting the system's performance and QoS. Improving response time and ensuring low-latency communication in cloud-based IoT settings are crucial concerns for delay-sensitive applications, including smart surveillance systems, autonomous vehicles, healthcare monitoring, and industrial IoT [7][8]. Conventional static resource allocation methods are not able to effectively manage dynamic workloads and busier network conditions, which leads to a greater response delay and lower QoS performance [9].

Predicting response times, understanding workload behaviour and dynamically allocating computational resources to minimise latency and optimise system efficiency are all achievable with ML and DL methods.

A major research goal is to optimize the response time: the faster the response time, the higher throughput and energy efficiency would be of the services provided in cloud-IoT applications, as well as other service attributes[10]. Deep learning models further enhance optimization ability by discovering complex connections between the network conditions, CPU utilization, memory usage and the way tasks are executed on the network [11][12]. This study is targeted to address the challenges presented and proposes an optimization framework for minimum response time for cloud-based IoT systems. One possible approach is to implement the ML and DL model to estimate the response time and dynamically distribute resources. An adaptive optimization mechanism dynamically adjusts CPU allocation and resource scheduling to reduce the response delay, increase the throughput and improve the resource scheduling with QoS-awareness in cloud-IoT environments[13][14]. The trial of the efficiency of the reaction time prediction and optimization for the IoT context using cloud computing demonstrates the usefulness of the proposed architecture. The main and key results of the study are:

- Proposed a comprehensive feature engineering mechanism based on workload dynamics, latency characteristics, and characteristic of resource interactions, which enables accurate prediction of response time.
- Built ML and DL based predictive models for intelligent response time estimation and adaptive optimization of dynamic IoT workloads.
- Architected an adaptive optimization engine that dynamically optimizes computational resources with workload conditions and network performance changes in real time.
- Improved latency management, resource utilization efficiency and adaptive system reliability in cloud-based IoT systems to enhance QoS.

The novelty of this work is in developing an adaptive optimization framework for cloud-based IoT systems that is latency-aware, incorporates machine learning, deep learning, and dynamic CPU allocation strategies, and optimizes for minimum response time. The designed framework combines response-time prediction, adaptive resource optimization,

© JGREC 2026, All Rights Reserved

throughput analysis and energy-efficiency evaluation into a single architecture unlike the conventional methods that only focus on the prediction accuracy. Using a DNN equipped with an adaptive optimization engine, intelligent decisions and resource utilization are improved in different IoT workloads. This need for effective, scalable, and low-latency cloud-IoT environments is driving research, because traditional static resource allocation methods fall short in meeting the quality of service and real-time requirements.

The rest of the paper is organized in this manner. Section II presents the pertinent work. Section III introduces adopting strategies. Section IV displays the experiment's findings. Finally, the results and future directions are highlighted in Section V.

II. Literature Review

This section presents a literature review on the use of data mining and machines for response time optimization in the cloud-IoT. Yaghoobi and Harandi (2026) implement the innovative QEAS framework, which merges DRL and quantum annealing. Adapting to network changes in real-time, the QEAS system represents task interdependencies using DAGs. So that latency-critical jobs are routed to the fog layer and resource overhead is reduced, the proposed technique dynamically distributes tasks to the cloud and fog layers. Under the CloudSim 5.0 simulated environment, QEAS is better than the base methods. The testing results demonstrate reduced energy consumption of 22%, reduced latency of 25% (with 94% of tasks under 10 ms), reduced operating expenses of 24%, and reduced migration overhead of 25% to 40%, in addition to optimizing resource utilization and ensuring no deadline violations [15].

Liu et al. (2025) use the Squirrel Search Algorithm (SSA) for virtual machine migration. These agents optimize load balancing and resource distribution by mimicking squirrel behavior during migration and search. The findings suggest that squirrel-driven approaches have the potential to address difficult problems in cloud computing settings. Through this approach, they can optimize the operations of their cloud infrastructure and enhance the reliability of their service operations and resource allocation. Using the suggested method, and able to decrease failures for most service providers by 12.12%, boost availability of virtual machines by 8.56 percent, and improve load mitigation by 11.59% [16].

Aldossary, Alharbi and Ayub (2024) use the IoT as its central focus to tackle this issue by suggesting efficient approaches to data placement and node management A real-time and accurate depiction of the data center's environment is made possible by the seamless integration of IoT data into the Hybrid TCN-GRU-NBeat (NGT) model. The NGT model can optimize the above-mentioned storage migration techniques using the latest data provided by the IoT sensors (SOA)[17].

Faruqui et al. (2024) Using Machine Vision at the IoT Edge (Mez) technology, a new technique and strategy dubbed GridSensing (GRS) is introduced to optimize resource use and minimize Cloud IaaS costs. A 22.43% reduction in storage requirements and a 31.29% reduction in bandwidth were both demonstrated by the experiment's outcomes. Various Internet of Things (IoT) nodes dynamically rate the current bandwidth requirements, which opens the door to the potential of implementing the GRS algorithm in an IoT camera-based security grid. This proposed idea also helps in reducing the overall throughput of the whole grid which results in significant optimization of cloud resources[18] .

Samuel, Vinothini and Kanniappan (2023) present a POA that uses Attention 1DCNN-LSTM architecture (POA-A1DCNN-LSTM) to maximise the utility of MEC servers by reducing power consumption and task delay. Then, tasks are offloaded using the A1DCNN-LST architecture, which selects the fog node with the lowest processing latency and shortest task duration. Experimental results demonstrated that the suggested approach achieved better performance than state-of-the-art approaches across multiple metrics: 0.5 j, 1.5 ms, 60 s, 385 ms, and 0.98 [19].

Aldossary (2023) proposes the MILP mathematical model as a viable optimization approach for identifying Internet of Things (IoT) applications within a multi-layer fog-cloud environment. Data transmission, power consumption, and cost are just a few of the numerous objectives that may be optimized for when using this technique to evaluate the needs of IoT applications, the capacity of available resources, and the geographic locations of servers. Results showed that the proposed solution cut processing and networking power consumption costs by up to 78% and data transmission costs by 64 percent compared to the existing cloud-based method. Ultimately, a heuristic method was created to verify and replicate the strategy that was provided. It produced outcomes that were similar to the suggested model, with a maximum difference of 5.4% of the overall power usage [20].

Research Gap: Despite significant progress in cloud–IoT response time optimization through diverse approaches such as quantum-enhanced scheduling, bio-inspired VM migration, hybrid deep learning models, and optimization heuristics, existing studies primarily focus on isolated aspects like energy efficiency, task offloading, or VM migration. Most frameworks are validated in synthetic or simulation-based environments (e.g., CloudSim), limiting their applicability in real-world heterogeneous IoT deployments. Moreover, these approaches can provide latency reduction and resource optimization, but they typically don't include adaptive real-time optimization engines that dynamically adapt to the stress variations between fog–edge–cloud layers. This poses a challenge to have an integrated solution which is capable of minimizing response time, meeting SLAs, and simultaneously scheduling energy efficiently in real-world multi-tier IoT systems using AI.

III. Methodology

The proposed method is based on building a minimum response time optimization framework using machine for intelligent resource scheduling and latency-aware optimization (Fig. 1). The data is subjected to Exploratory Data Analysis (EDA) after undergoing preprocessing and enhancement using Feature Engineering. The data obtained is normalized and split into training and testing sets. RF, GB, SGD, and DNN have been trained and evaluated using machine learning models, with DNN exhibiting the lowest prediction error. Finally, a latency-aware adaptive optimization engine dynamically adjusts CPU allocation and workload scheduling to minimize response delay, improve throughput, and enhance QoS performance in dynamic cloud-IoT environments. Detailing of propose flowchart steps are given below:

© JGREC 2026, All Rights Reserved

A. *Data Collection*

Data collection is the first phase that is employed to collect information for model training and analysis. This work uses data from the Kaggle Multi-Tier IoT Resource Allocation Dataset [21] consisting of 1000 data samples with 13 features. The dataset includes parameters such as CPU usage, memory usage, network latency, task execution time, workload type, and resource allocation. These attributes give information about system performance of IoT systems and used for response time prediction and Adaptive Optimization.

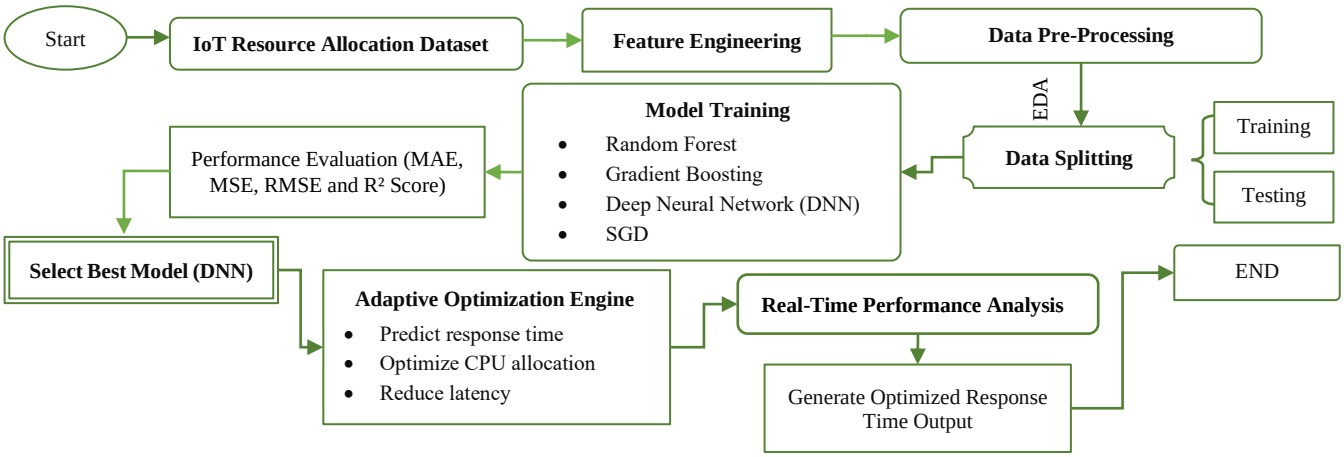

Fig. 1. Proposed Flowchart for Response Time Optimization

B. *Feature Engineering & Pre-Processing*

Feature engineering and pre-processing are carried out to ensure a high-quality data set and the quality of the predictions. A total of 9 new features are generated from the original dataset, including Temperature, Humidity, Allocation Error, Total Latency, Response Time Efficiency, CPU–Memory Interaction, Latency–Jitter Ratio, IsWeekend, and IsPeakHour, which help capture workload behavior, latency variations, and resource allocation patterns. These features designed to enhance the prediction of the cloud-based IoT response time by capturing the workload patterns, latency variations and the behavior of resource allocation in cloud-based IoT systems. During pre-processing, the dataset is examined for inconsistencies, resulting in 0 missing values and 0 duplicate records, ensuring clean and reliable data.

C. *Exploratory Data Analysis (EDA)*

EDA is conducted to study the data patterns, feature distributions, and relationships between the IoT parameters. It helps to understand workload behavior and it aids in the effective development of models.

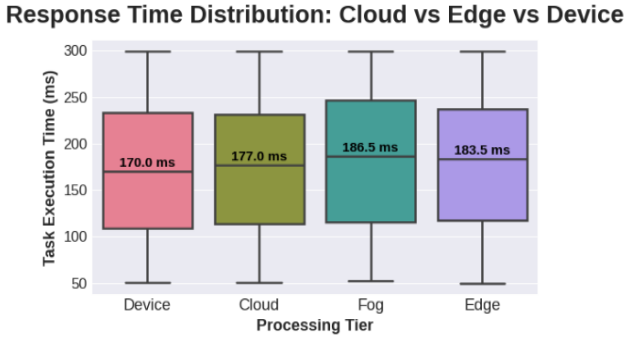

Fig. 2. BoX Plot For Response Time Distribution Across Processing Tiers

The execution time distribution of the tasks per tier is shown in the box plots of Fig. 2. The latency value grows slightly from 170.0 ms at the Device level to 186.5 ms at the Fog tier due to distributed processing. Similar variation is observed for both Cloud and Edge tiers implying equal performance between centralized and near-sourced computing. The visualization overall highlights the effects of architectural tiering on responsiveness and efficiency in multi-tiers computing environments.

The task execution time is plotted in Fig. 3 as a function of frequency, ranging from 50.0 ms to 299.0 ms. The distribution is almost symmetrical with only a slight negative skewness of -0.06 and a very similar central tendency with a Mean of 175.9 ms (red dashed line) and a Median of 179.5 ms (orange solid line). The dispersion around the mean is highlighted by a light pink shaded band which extends for about 104.4 ms to 247.4 ms, within one standard deviation, and the peak frequency counts get close to 34 occurrences for individual latency bins.

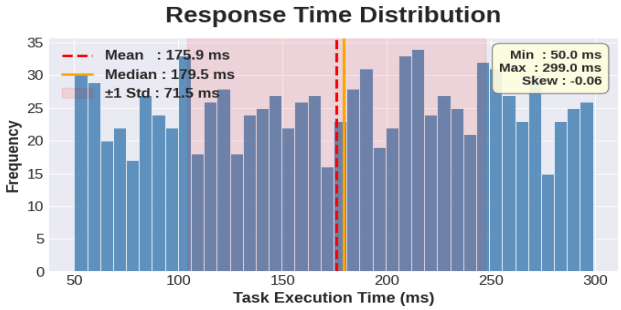

Fig. 3. Histogram for Response Time Distribution

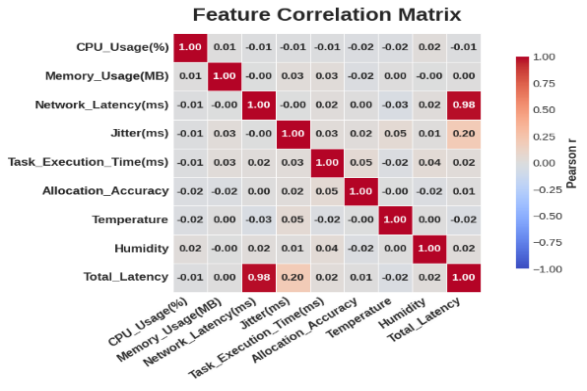

Fig. 4. Correlation Matrix of features

© JGREC 2026, All Rights Reserved

The result of the key system performance features including CPU, memory usage, network latency, jitter and total latency are displayed as Pearson correlation coefficient in Fig. 4. There are high positive correlations between network latency and total latency (r = 0.98), showing that they are closely related to one another and affect overall latency. The close to zero correlation of most other feature pairs imply a low linear correlation. The heatmap clearly displays significant dependencies, helping to pinpoint critical parameters for optimization and resource allocation methods.

D. Data Splitting and Normalization

The data set is divided into 80% training set and 20% testing set to train and evaluate model. After the splitting, normalization using 'standardisable' is used to scale the features and make variance more compact, which is conducive to learning. These equations can be written as Equation (1) for the normalisation process:

$$Z = \frac{X-\mu}{\sigma} \quad (1)$$

Where X denotes the OriginalValue, μ is the mean, and σ is the standard deviation of the feature.

E. Model Selection

Multiple models including RF, GB, DNN, and Stochastic Gradient Descent (SGD) are selected and trained to predict response time in cloud-based IoT systems. Using normalized training data, the models are trained and their performance is compared to determine the most successful technique.

1) Random Forest

RF is an EL method that combines DT with other techniques to produce more accurate and robust predictions. By randomly selecting features and training data, RF builds a DT network. Each tree in the forest uses a distinct data sample to train its predictions; the final forecast is the result of combining all of these guesses.

2) Gradient Boosting

GB is an Ensemble Learning method that combines several ineffective models into a more robust prediction. The basic idea behind GB is to construct a model, then use it to find mistakes and train another model to fix them [22]. The whole process is repeated with a new model, and the prediction improves in the model until it is accurate enough.

3) Deep Neural Network (DNN)

A DNN model uses a number of Hidden Layers to represent the nonlinear, complicated connections between characteristics in the input and the predictions made from the output. The layers are interconnected and encode hierarchical representations and progressively discover patterns in high dimensional data. The model employs backpropagation and optimization methods like Adam to reduce prediction errors, thereby improving the model's generalization ability. The overall performance of the DNN architecture is good in terms of predictive power for the tasks with complex feature interactions and massive data sets.

4) Stochastic Gradient Descent (SGD)

SGD is a popular optimization technique for ML models. SGD method iteratively adjusts the model's parameters to reduce the loss function. One segment of the data is sampled at random and the method alters the parameters of the model by computing a gradient on this sample. Scalability, rapid convergence, and efficiency are three of SGD's many benefits.

Model training is performed using RF, GB, DNN, and Stochastic Gradient Descent (SGD) models to predict response time in cloud-based IoT systems. Feature selection using square roots, a minimum sample split of 5, and 300 estimators are used to train the Random Forest model. The maximum depth is 20. With a Learning Rate of 0.05, a maximum depth of 8, and a subsample ratio of 0.8, the Gradient Boosting model is used. The convolutional neural network (CNN) model has 100 iterations, several hidden layers (256, 128, 64, and 32 neurons), ReLU activation, dropout regularization (0.4, 0.3, 0.2), Adam Optimizer (0.001 learning rate), Batch Normalization, and so on. The SGD model is trained with an adaptive Learning Rate, initial Learning Rate of 0.01, and BatchSize of 50. Model performance is evaluated to identify the best predictor, where the DNN model achieves the lowest prediction error.

F. Regression Evaluation Metrics

The proposed models were evaluated in this research using a variety of metrics. The forecasting mistake known as the MSE is calculated by squaring and averaging the two numbers to determine the difference between the actual and projected values. The square root of the MSE is called the RMSE. The fact that it squares the MSE makes it less mistake-size sensitive. A weighted average of all absolute errors is known as the MAE. In comparison to MSE, it is less affected by outliers and responds less strongly to the extent of the prediction error. On the other hand, is a statistical measure that illustrates the extent to which the independent variables in a model can explain the dependencies among themselves. From 0 to 1, where 1 signifies a perfect match and 0 no variance described by the model. Equations (2)-(5) are as follows:

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

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (3)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} | y_i - \hat{y}_i | \quad (4)$$

$$R^2 = 1 - \frac{SSE}{SST} \quad (5)$$

Where, n stands for the sum of all observations, y_i denotes the actual value, $\hat{y}_i$ indicates the predicted value.

IV. Results and Discussion

The tests were performed on a Windows 10 machine with a 64-bit Intel(R) Core(TM) i7-7500U CPU and 16 GB of RAM. These models have been built using the PyTorch v1.9.0 module in Python v3.8.2. Table I presents the regression performance comparison of propose models for response time prediction. Among all models, the DNN model achieves the best performance with the lowest MAE, MSE, and RMSE, along with the highest R^2 score (0.9830), indicating superior prediction accuracy and better model fitting. The GB model also yields a good accuracy of R^2=0.9630 and RF gives moderate accuracy. The error values are the highest in the SGD model, and the R^2 value is the lowest (0.7760), which means that it has a weak prediction ability. The findings demonstrate that among the models tested, the DNN model achieves the best results for optimizing adaptive response times in cloud-based IoT systems.

TABLE I. Regression Performance Analysis of Machine Learning Models

Model	MAE (ms)	MSE (ms^2)	RMSE (ms)	R^2 Score

© JGREC 2026, All Rights Reserved

RF	19.7023	603.9137	24.5747	0.8865
GB	10.2383	196.5430	14.0194	0.9630
DNN	5.9605	58.3128	7.6630	0.9830
SGD	28.9468	1191.5817	34.5193	0.7760

Fig. 5 shows the expected response time variation for a data analytics device with different CPU allocations. The blue curve represents the entire range of predicted response times, with the best CPU allocation being 10%, and the lowest response time being 184.7 ms. The current configuration at 24% CPU has a slightly higher response time of 187.4 ms. The adaptive optimization shows performance improvement of 1.44%.

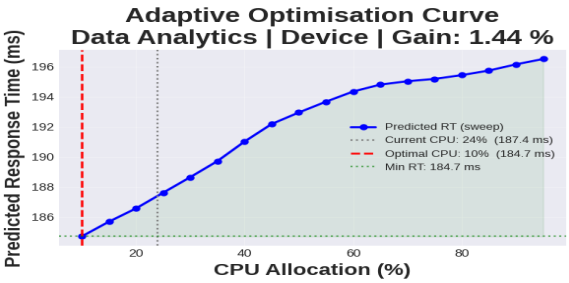

Fig. 5. Adaptive Optimization Curve for CPU Allocation

Fig. 6. Training Dynamics of Deep Neural Network

Fig. 6 shows how the training and validation metrics change with different epochs of training a deep neural network. The MSE loss in the left figure drops sharply within the first 10 epochs, then levels off, which suggests that the network can learn and converge effectively. The MAE plot shows a similar behavior with both training and validation curves closely following each other as shown in the right plot. These plots indicate good generalization and no overfitting in model training.

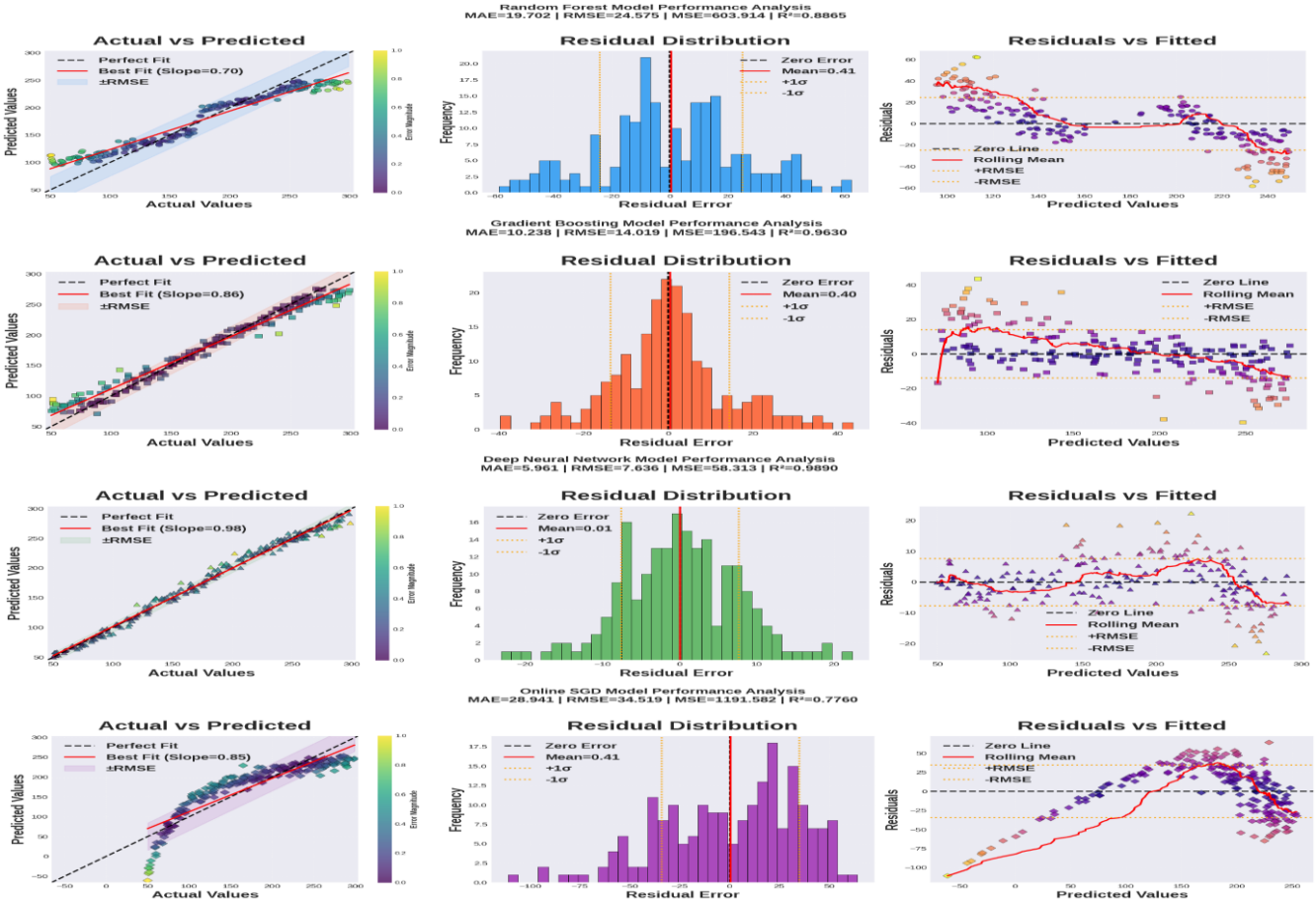

Fig. 7. Actual Vs. Predicted and Residual Analysis of Machine Learning Models

A comparative study of four ML models is shown in Fig. 7. The Actual vs Predicted graphs show the accuracy of the predictions by each model, with the DNN model having the closest relation with the Ideal fit line, which means that it has better predictive ability. The residual distribution plots show the errors around zero, with DNN having the most balanced and compact distribution, indicating lower prediction errors. The Residuals vs Fitted plots then further investigate errors uniformity, and reveal nonlinear patterns and larger residuals variation in Random Forest and Online SGD, while relatively stable behavior in DNN and Gradient Boosting. The DNN model has the highest reliability and error measure in prediction

© JGREC 2026, All Rights Reserved

The analysis for response time and throughput shows that all the models tested give almost the same average actual response time and only small differences in the predicted response time (see Table II). This leads to marginally better efficiency in terms of DNN having the lowest predicted task completion time (177.25 s) and the highest throughput (5.6418 tasks). Table III shows a comparison of the latency and jitter performance of models. The DNN model has the lowest latency reduction of −0.01%, meaning that the DNN model improves optimization of response. The mean network latency (26.82 ms) and P95 network latency (47 ms) are consistent across all models indicating that the network is stable. The overall performance of DNN model shows better latency optimization with stable network performance.

TABLE II. RESPONSE TIME AND THROUGHPUT ANALYSIS OF EVALUATED MODELS

Metric	RF	GB	DNN	SGD
Avg Actual Response Time (ms)	177.2400	177.2400	177.2400	177.2400
Avg Predicted Response Time (ms)	177.6500	177.6300	177.2500	177.6500
Task Completion Time (s)	177.6500	177.6300	177.2500	177.6500
Throughput (tasks/s)	5.6292	5.6296	5.6418	5.6292
Exec. Time Reduction (%)	-0.2300	-0.2200	-0.0100	-0.2300

TABLE III. LATENCY AND JITTER PERFORMANCE COMPARISON OF MACHINE LEARNING MODELS

Metric	RF	GB	DNN	SGD
Latency Reduction Rate (%)	-0.23	-0.22	-0.01	-0.23
Mean Network Latency (ms)	26.82	26.82	26.82	26.82
P95 Network Latency (ms)	47.00	47.00	47.00	47.00
Mean Jitter (ms)	5.01	5.01	5.01	5.01
Jitter Std (ms)	2.63	2.63	2.63	2.63
Jitter Range (ms)	8.00	8.00	8.00	8.00

TABLE IV. RESOURCE UTILIZATION ANALYSIS OF MACHINE LEARNING MODELS

Metric	RF	GB	DNN	SGD
CPU Utilisation (%)	49.4600	49.4600	49.4600	49.4600
Memory Utilisation (MB)	4476.2500	4476.2500	4476.2500	4476.2500
Resource Allocation Eff. (%)	97.4700	97.4700	97.4700	97.4700
Resource Consumption Rate	22.2188	22.2188	22.2188	22.2188
Load Balancing Efficiency	0.248793	0.248793	0.248793	0.248793

TABLE V. QOS AND RELIABILITY PERFORMANCE COMPARISON

Metric	RF	GB	DNN	SGD
QoS Improvement (%)	-0.2300	-0.2200	-0.0100	-0.2300
SLA Compliance Accuracy (%)	94.0000	98.0000	96.0000	92.0000
SLA F1-Score	0.9459	0.9831	0.9649	0.9266
Error Rate (%)	6.0000	2.0000	4.0000	8.0000
System Reliability (%)	100.0000	100.0000	100.0000	100.0000

Tables IV and V compare resource utilization, QoS, and reliability performance of the ML models. Resource metrics such as CPU utilization (49.46%), memory utilization (4476.25 MB), and resource allocation efficiency (97.47%) remain similar across all models, indicating stable resource management. For QoS performance, the GB model achieves the highest SLA compliance accuracy, best F1-score (0.9831), and lowest error, while maintaining 100% system reliability. The DNN model is capable of providing better adaptive response optimization of QoS, with the QoS improvement value of −0.01%.

TABLE VI. EFFICIENCY AND OPTIMISATION METRICS OF EVALUATED MODELS

Metric	RF	GB	DNN	SGD
Prediction Accuracy (%)	29.0000	57.5000	74.5000	13.0000
Energy Efficiency Index (%)	0.001023	0.001968	0.003300	0.000696
Task Scheduling Efficiency	4.7079	8.6725	14.0025	3.2553
Adaptive Opt. Efficiency (%)	88.8000	94.2200	96.6400	83.6700
Performance Gain (%)	88.8000	94.2200	96.6400	83.6700

TABLE VII. MODEL TIMING PERFORMANCE ANALYSIS

Metric	RF	GB	DNN	SGD
Training Time (s)	1.465990	1.058700	36.048300	0.035200
Inference Time (ms/sample)	0.582112	0.030863	1.442076	0.001876

Tables VI and VII show the trade-off between the models' optimization performance and the computation time. Energy efficiency, response time optimization ability, and prediction accuracy are the highest for DNN model, showing that DNN model has better capability for the response time optimization, and it is a strong overall performance. However, this improved performance results in increased training (36.04s) and inference time (1.44 ms/sample). On the other hand, the SGD model has the fastest running speed while it has not the best optimization effect. The results show that DNN focuses on optimizing prediction quality and efficiency over computational speed, whereas SGD focuses on computational speed.

TABLE VIII. SAMPLE REAL-TIME OPTIMIZATION OUTPUT

Parameter	Value
best_model_used	DNN
current_predicted_RT_ms	979.74
optimal_cpu_allocation_%	10
minimum_achievable_RT_ms	979.72
improvement_%	0.0
recommended_cpu_boost_%	5
sla_compliant_200ms	False
throughput_tasks_per_sec	1.021
energy_efficiency_index	0.003728

The results in Table VIII illustrate the success of the DNN based real-time optimization framework for response time prediction and resource allocation. The model is able to find the best CPU assignment with 10% CPU and recommends another 5% CPU boost to enhance performance in high latency scenarios. While some improvement in response time is gained, the system still operates at a moderate energy efficiency with a throughput of 1.021 tasks/sec. It is also observed from the results that the optimized response time is still greater than the SLA of 200ms, which further suggests the need for developing more efficient strategies to achieve further reduction in latency.

A. Comparative Analysis and Discussion

The performance of the prediction of existing models and the proposed DNN model is compared using the metrics in Table IX. The proposed DNN model has the highest R^2 value

© JGREC 2026, All Rights Reserved

which shows the highest model fitting and prediction capability. The CatBoost model yields a MAE of 3.057 and an R^2 score of 0.95, which is quite good for its prediction ability, while the RF model has a lower MAE of 2.74 and RMSE of 4.46. The R2 value of the CNN model is 0.968, and the ANN model has an MAE of 1.36, which indicates that the model has a lower prediction error. The results indicate that despite some models yielding lower error scores during the baseline, the DNN model offers better overall prediction accuracy in terms of reliability and optimization of response time for cloud IoT systems.

TABLE IX. COMPARATIVE PERFORMANCE ANALYSIS OF EXISTING MODELS AND PROPOSED MODEL

Model	MAE	MSE	RMSE	R^2 Score
Cat boost [23]	3.057	45.69	6.76	0.95
RF[24]	2.74	19.89	4.46	0.82
CNN [25]		0.008		0.968
ANN [26]	1.36	-	2.15	-
MLP [27]	-	-	0.326	-
DNN	5.9605	58.3128	7.6630	0.9830

The proposed minimum response time optimization framework can be beneficial for cloud-based IoT systems in several ways: prediction of response time, adaptive CPU allocation, minimization of latency, maximization of throughput, and maximization of energy efficiency. The DNN model is applied to improve the accuracy of prediction and optimization performance and the adaptive characteristic should optimize better throughput and resource utilization. Furthermore, the framework can enable intelligent decision making and improve on the QoS and reliability of the system in dynamic IoT environments.

V. CONCLUSION AND FUTURE WORK

Cloud-based IoT systems have grown rapidly, adding new obstacles to the issues of high response time, low efficiency of resource allocation, and optimization of latency, which can impact system performance and Quality of Service (QoS). This research presents a minimum response time optimization framework by using ML/DL model to address the aforementioned issues. The framework is responsible for feature engineering, preprocessing, model training, and adapting the model to optimize the prediction of response times and intelligent resource allocation. Results of the experiments demonstrate that the model of DNN has R^2 value equal to 0.9830, which increases the accuracy of the model, improves the efficiency of optimization, and strengthens the reliability of the system. The proposed solution also enhances cloud-based IoT systems' energy efficiency, throughput and adaptive resource management. There are some limitations in the study, however. The framework is evaluated on one benchmark dataset, which might not be representative of actual IoT scenarios. Further, the optimized response time does not meet strict SLAs latency requirements, suggesting room for further improvement. Future research could involve the integration of real-time datasets from the IoT with cloud-to-edge systems and more sophisticated methods like reinforcement learning and federated learning to enhance scalability, latency, and efficient adaptive optimization.

REFERENCES

[1] A. Warrier, "iPaaS Solutions for Healthcare Enterprise Integration: Cloud-Native Integration Platforms for Multi-System Orchestration," *Int. J. Lead. Res. Publ.*, vol. 3, no. 1, pp. 1–9, Jan. 2022, doi: 10.70528/IJLRP.v3.i1.1770.

[2] P. Kumar, "Edge Computing and IoT for Real-Time Healthcare Data Processing and Integration," in *2025 4th International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, IEEE, Dec. 2025, pp. 105–110. doi: 10.1109/ICAAIC64647.2025.11331211.

[3] U. K. Padyana, H. P. Rai, P. Ogeti, N. S. Fadnavis, and G. B. Patil, "AI and Machine Learning in Cloud-Based Internet of Things (IoT) Solutions: A Comprehensive Review and Analysis," *Integr. J. Res. Arts Humanit.*, vol. 3, no. 3, pp. 121–132, May 2023, doi: 10.55544/ijrah.3.3.20.

[4] V. K. Sharma, "Cloud Computing IoT: 5G Focused IoT with Cloud Solutions," *Int. J. AI, BigData, Comput. Manag. Stud.*, vol. 6, no. 3, 2025, doi: 10.63282/3050-9416.IJAIBDCMS-V6I3P103.

[5] I. Ficili, M. Giacobbe, G. Tricomi, and A. Puliafito, "From Sensors to Data Intelligence: Leveraging IoT, Cloud, and Edge Computing with AI," *Sensors*, vol. 25, no. 6, p. 1763, Mar. 2025, doi: 10.3390/s25061763.

[6] N. D. Bhandarwar, "A Mathematical Framework for Explainable and Adversarially Robust IDS Using ML for Large-Scale Enterprise and Cloud Systems," *Int. J. Appl. Math.*, vol. 39, no. 1, 2025.

[7] O. Almurshed *et al.*, "Enhancing performance of machine learning tasks on edge-cloud infrastructures: A cross-domain Internet of Things based framework," *Futur. Gener. Comput. Syst.*, vol. 166, p. 107696, May 2025, doi: 10.1016/j.future.2024.107696.

[8] Y. Patel, "Self-Adaptive Al-Based Orchestration for Multi-Cloud Interoperability and Performance Optimization," in *SoutheastCon 2026*, Huntsville, AL, USA: IEEE, 2026, pp. 01–08, April. doi: 10.1109/SoutheastCon63549.2026.11476031.

[9] K. Jangiti, "Design and Validation of a Machine Identity Governance Framework for AI Agents in Multi-Cloud Environments," in *SoutheastCon 2026*, IEEE, Feb. 2026, pp. 1–6. doi: 10.1109/SoutheastCon63549.2026.11476363.

[10] C. Patel, "A Review of Multi-Channel CRM Strategies Using Big Data and Cloud Integration," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 8, no. 1, pp. 577–588, 2022.

[11] V. Sanikal, "AI-Enhanced Network Security Framework for Cloud-Edge Integrated Environments," in *2026 Innovations in Machine, Engineering, and Digital Conference (IMED)*, 2026, pp. 1–6. doi: 10.1109/IMED68921.2026.11484417.

[12] S. Singh, "Performance Evaluation of Machine Learning Regression Models for 5G Network Resource Allocation Optimization," in *2025 International Conference on Multimedia Computing, Networking and Applications (MCNA)*, 2025, pp. 47–52. doi: 10.1109/MCNA65829.2025.11124374.

[13] B. Krishnan, S. Perla, S. Maddela, and R. Lingam, "Adaptive Multi-Cloud Infrastructure for CRM Analytics: Real-Time ML and Data Sync with LLMs," in *2025 IEEE 3rd Global Conference on Wireless Computing and Networking (GCWCN)*, IEEE, Nov. 2025, pp. 1–8. doi: 10.1109/GCWCN66157.2025.11448404.

[14] B. P. Singh and H. Singh, "Using LLMs for Autonomous Cloud Infrastructure Entitlement Management to Prevent Overprivileged Access," *J. Eng. Comput. Sci.*, vol. 5, no. 4, pp. 1–14, April, 2026, doi: https://doi.org/10.5281/zenodo.19488212.

[15] A. Yaghoobi and M. N. Harandi, "Quantum-enhanced adaptive scheduling for real-time IoT task optimization in fog-cloud systems," *Comput. Networks*, vol. 278, p. 112088, Apr. 2026, doi: 10.1016/j.comnet.2026.112088.

[16] C. Liu, L. Ma, M. Zhang, and H. Long, "Optimizing cloud resource management with an IoT-enabled optimized virtual machine migration scheme for improved efficiency," *J. Netw. Comput. Appl.*, vol. 237, p. 104137, May 2025, doi: 10.1016/j.jnca.2025.104137.

[17] M. Aldossary, H. A. Alharbi, and N. Ayub, "Exploring Multi-Task Learning for Forecasting Energy-Cost Resource Allocation in IoT-Cloud Systems," *Comput. Mater. Contin.*, vol. 79, no. 3, pp. 4603–4620, 2024, doi: 10.32604/cmc.2024.050862.

[18] N. Faruqui *et al.*, "Cloud IaaS Optimization Using Machine Vision at the IoT Edge and the Grid Sensing Algorithm," *Sensors*, vol. 24, no. 21, p. 6895, Oct. 2024, doi: 10.3390/s24216895.

[19] P. Samuel, A. Vinothini, and J. Kanniappan, "POLSTM: Poplar optimization-based long short term memory model for resource allocation in cloud environment," *Comput. Commun.*, vol. 211, pp. 11–23, Nov. 2023, doi: 10.1016/j.comcom.2023.08.008.

© JGREC 2026, All Rights Reserved

[20] M. Aldossary, "Multi-Layer Fog-Cloud Architecture for Optimizing the Placement of IoT Applications in Smart Cities," *Comput. Mater. Contin.*, vol. 75, no. 1, pp. 633–649, 2023, doi: 10.32604/cmc.2023.035414.

[21] Ziya, "Multi-Tier IoT Resource Allocation Dataset," Kaggle.

[22] A. V. Konstantinov and L. V. Utkin, "Interpretable machine learning with an ensemble of gradient boosting machines," *Knowledge-Based Syst.*, vol. 222, p. 106993, Jun. 2021, doi: 10.1016/j.knosys.2021.106993.

[23] A. Pardeshi, "Hybrid Machine Learning Model for Auto Scaling using CPU Utilization," National College of Ireland, 2023.

[24] V. Krishna, "Comparative Analysis of Machine Learning Models for Predictive Resource Scheduling in Cloud Environments," 2025.

[25] M. Ali *et al.*, "A Machine Learning Approach to Reduce Latency in Edge Computing for IoT Devices," *Eng. Technol. Appl. Sci. Res.*, vol. 14, no. 5, pp. 16751–16756, Oct. 2024, doi: 10.48084/etasr.8365.

[26] V. K. Prasad, D. Dansana, M. D. Bhavsar, B. Acharya, V. C. Gerogiannis, and A. Kanavos, "Efficient Resource Utilization in IoT and Cloud Computing," *Information*, vol. 14, no. 11, p. 619, Nov. 2023, doi: 10.3390/info14110619.

[27] R. Zhang, L. Liu, M. Dong, and K. Ota, "On-Demand Centralized Resource Allocation for IoT Applications: AI-Enabled Benchmark," *Sensors*, vol. 24, no. 3, p. 980, Feb. 2024, doi: 10.3390/s24030980.

© JGREC 2026, All Rights Reserved