International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

AI-Driven Edge Computing for Intelligent Resource Management in Internet of Things Networks

Mr. Aimn Azoumi Mohammed Ikriam

Academic, Information Technology, Higher Institute of Science and Technology -Wadi-Aial- Libya

Abstract

The rapid growth of IoT applications has increased the demand for efficient computational resources. Edge computing reduces latency by processing data closer to IoT devices, but limited and heterogeneous edge resources make task management challenging. This study proposes an AI-driven resource management framework for intelligent task allocation in IoT edge environments. The framework analyzes workload and resource conditions, including CPU utilization, memory, latency, bandwidth, and task requirements, to select suitable edge nodes. The proposed approach is evaluated using execution time, latency, energy consumption, resource utilization, task completion rate, and throughput. The study aims to demonstrate that AI-based resource management can improve resource utilization, reduce latency and energy consumption, and enhance the performance of IoT edge computing systems.

Keywords: Internet of Things, Edge Computing, Artificial Intelligence, Machine Learning, Resource Management, Task Offloading, Resource Allocation, Latency, Energy Efficiency.

1. Introduction

The Internet of Things (IoT) has become an important computing paradigm that enables physical devices to collect, process, and exchange data through communication networks. IoT environments include sensors, smart home appliances, wearable devices, industrial equipment, vehicles, healthcare systems, and environmental monitoring devices [1]. The continuous growth in the number of connected devices has resulted in a substantial increase in the volume of generated data and the computational requirements associated with processing this data [2]. Traditional cloud computing architectures can provide large-scale computational and storage resources; however, transferring large amounts of IoT data to remote cloud servers can introduce considerable network latency, bandwidth consumption, and energy overhead [3].

Edge computing has emerged as an effective approach for addressing some of these limitations by bringing computational and storage resources closer to IoT devices. Instead of sending all generated data to centralized cloud servers, computational tasks can be processed by geographically distributed edge nodes located near data sources. This architecture can significantly reduce communication delays and improve the responsiveness of latency-sensitive IoT applications. Edge computing is particularly beneficial for applications such as smart healthcare, autonomous transportation, industrial automation, smart cities, and real-time surveillance, where rapid data processing is essential [4].

Despite these advantages, efficient resource management remains a significant challenge in edge computing environments. Unlike centralized cloud data centers, edge nodes generally have limited and

heterogeneous computational resources. Different edge nodes may have different CPU capacities, memory sizes, available bandwidth, processing speeds, and energy constraints. In addition, IoT workloads are highly dynamic because the number of connected devices and the amount of generated data can change continuously [5]. Consequently, static resource-allocation strategies may result in inefficient resource utilization, increased task execution time, and unnecessary energy consumption [6].

Task offloading is one of the important mechanisms used to address these challenges. Through task offloading, computationally intensive tasks generated by IoT devices can be transferred to suitable edge nodes for processing [7]. However, selecting an appropriate edge node is a complex decision because multiple factors must be considered simultaneously. A node with sufficient computational capacity may have high network latency, while another node may have low latency but insufficient CPU or memory resources. Therefore, intelligent decision-making mechanisms are required to dynamically select appropriate resources according to current network and computational conditions [8].

Artificial Intelligence (AI) and Machine Learning (ML) techniques provide promising solutions for intelligent resource management in dynamic computing environments. Machine learning models can learn relationships between workload characteristics, resource availability, network conditions, and task execution performance. Such models can subsequently support automated resource-allocation decisions without relying exclusively on predefined rules. AI-driven approaches can therefore provide more adaptive resource management compared with conventional static or rule-based techniques [9].

However, existing resource-management approaches frequently face challenges related to dynamic workloads, heterogeneous edge resources, computational overhead, and the need to balance multiple performance objectives. Furthermore, achieving low latency while maintaining efficient energy consumption and high resource utilization remains a challenging optimization problem. These challenges motivate the development of intelligent resource-management mechanisms capable of adapting to changing IoT workloads and edge-network conditions [10].

This study proposes an **AI-driven edge computing framework for intelligent resource management in IoT networks**. The proposed framework collects information about IoT tasks and available edge resources, including CPU utilization, memory utilization, network latency, bandwidth availability, task size, computational requirements, and queue length [11]. These parameters are processed by an AI-based decision mechanism to identify suitable edge resources for incoming computational tasks. The framework is designed to improve task allocation efficiency while reducing latency and energy consumption and improving overall resource utilization.

2. Related Work

2.1 Edge Computing for IoT

Discuss:

- Cloud limitations for IoT
- Edge computing architecture
- Benefits of processing close to IoT devices
- Latency and bandwidth reduction
- Resource limitations at edge nodes

2.2 Resource Management in Edge Computing

Discuss:

- Resource allocation

- Task scheduling
- Task offloading
- Load balancing
- Energy-aware resource management

2.3 AI and Machine Learning for Edge Resource Management

Discuss:

- Machine learning-based resource prediction
- Deep learning
- Reinforcement learning
- Intelligent task offloading
- AI-based load balancing

2.4 Research Gap

This is particularly important. We should establish that many existing approaches optimize **one or two objectives**, while our proposed framework considers a combination of:

Latency + Energy Consumption + Resource Utilization + Task Completion + Throughput

and uses multiple real-time resource indicators to make the allocation decision.

2.5 Edge Computing for IoT

The increasing number of IoT devices and the volume of data generated by these devices have created significant computational and communication challenges for centralized cloud architectures. Edge computing addresses these limitations by moving computational resources closer to IoT devices, thereby reducing communication distance and potentially improving latency and responsiveness. However, edge nodes generally have more constrained and heterogeneous resources than centralized cloud servers, making resource management an important research problem. Recent surveys identify resource allocation, workload balancing, task scheduling, resource provisioning, and quality-of-service management as major challenges in cloud, fog, and edge environments [12].

In IoT environments, computation offloading allows resource-constrained devices to transfer computational tasks to nearby edge servers. This approach can reduce the processing burden on end devices and improve application response time. A 2024 comprehensive review of IoT task offloading in multi-access edge computing (MEC) classified existing approaches according to their mechanisms, optimization objectives, evaluation methods, and supported parameters, while also identifying continuing challenges in efficient offloading decisions [13].

2.6 Resource Management and Task Offloading

Efficient task offloading is particularly challenging because edge environments are dynamic and heterogeneous. An offloading decision must consider factors such as task size, computational requirements, available CPU resources, queue length, communication conditions, and latency requirements.

For example, a 2024 study proposed an optimized task-offloading strategy in an IoT edge-computing network in which an edge orchestrator determines whether tasks should be processed locally, at an edge server, or in the cloud. The study incorporated maximum tolerable latency into the decision process and reported improvements in task completion, execution time, network communication, and energy consumption [14].

Another 2024 study introduced the **Balanced Offload for Multi-type Tasks (BOMT)** approach, which prioritizes tasks according to characteristics such as task type, task size, and maximum tolerable delay

while considering the current MEC-server load. Simulation results indicated improvements in system delay, user coverage, and task completion rates [15].

These studies demonstrate the importance of considering both **task characteristics and current edge-resource conditions** when making allocation decisions.

2.7 Artificial Intelligence for Edge Resource Management

Traditional optimization and rule-based approaches can become difficult to apply when the number of IoT devices and edge nodes increases and when resource conditions change dynamically. Artificial Intelligence and Machine Learning provide an alternative because models can learn relationships between system conditions and resource-management decisions.

Recent research has increasingly investigated reinforcement learning and deep learning for dynamic task offloading and resource allocation. For example, a 2024 study combined predictive analytics with reinforcement learning to predict resource availability before making task-offloading decisions. The reinforcement-learning component then selected an appropriate offloading location and optimized resource-allocation policies. The authors reported improved resource utilization and reduced resource idle time compared with non-predictive approaches [16].

Similarly, a 2024 study proposed a **Decision Tree Empowered Reinforcement Learning (DTRL)** framework for computational offloading and resource allocation in IoT applications. The approach was designed to address dynamic offloading decisions, heterogeneous resources, and workload imbalance. Its simulation results showed improvements in delay, energy consumption, waiting time, task acceptance ratio, and service cost [17].

Deep learning has also been applied to task-offloading decisions. A 2024 study developed a CNN-LSTM-attention-based approach for cloud-edge-end collaboration, using a combined objective involving execution delay and energy consumption. This demonstrates the growing interest in using deep learning to model complex relationships within dynamic edge-computing environments [18].

Another recent direction involves combining resource management with energy-aware task offloading. The **EDITORS** framework, published in 2024, employed deep reinforcement transfer learning for energy-efficient dynamic task offloading in SDN-enabled edge nodes while also considering trust management [19].

2.8 Comparison of Recent Studies

Table 1. Recent Studies Comparison

Study	Main Approach	Main Focus	Evaluation	Limitation/Opportunity
2024 IoT task-offloading review	Review	MEC task offloading	Literature analysis	Identifies need for improved adaptive approaches [20]
Optimized task offloading, 2024	Edge orchestrator + scheduling	Latency, energy, execution time	Simulation	Relies primarily on optimization/scheduling
Predictive analytics + RL, 2024	Prediction + RL	Resource availability and offloading	Experimental evaluation	More complex than conventional ML approaches [21]

Study	Main Approach	Main Focus	Evaluation	Limitation/Opportunity
DTRL, 2024	Decision Tree + RL	Offloading and resource allocation	Simulation	RL training/complexity remains a consideration [22]
CNN-LSTM-Attention, 2024	Deep learning	Cloud-edge-end offloading	Simulation	Deep architecture can introduce computational overhead [23]
EDITORS, 2024	Deep RL + transfer learning	Energy-aware offloading	Simulation	Focuses strongly on energy/trust dimensions [24]
iMTOSA, 2025	Intelligent scheduling	Resource sharing/task allocation	Simulation	Focused on specific intelligent scheduling scenarios [25]

3. Research Gap

Based on the recent literature, several important observations can be made.

First, existing studies have demonstrated that AI and reinforcement learning can improve task offloading and resource allocation in edge environments. However, many approaches concentrate on a particular optimization objective, such as latency, energy consumption, or task completion. Other studies employ sophisticated deep reinforcement-learning architectures, which can introduce additional training and computational complexity [26].

Second, edge-resource management is inherently **multi-dimensional**. CPU utilization, memory availability, bandwidth, network latency, workload, task size, queue length, and energy consumption can simultaneously influence the quality of an allocation decision. A resource-management framework that considers these variables jointly can potentially provide more balanced decisions than approaches optimized primarily around a single objective.

Third, recent research demonstrates the value of **predictive intelligence** in edge resource management. Predicting resource availability before making an offloading decision can reduce resource idle time and improve utilization. [27] This creates an opportunity to investigate a comparatively lightweight ML-based framework that combines resource-state prediction with intelligent task allocation.

4. Proposed Research Contribution

Based on this gap, I recommend that we **do not try to compete with the most complicated Deep Reinforcement Learning models**. Instead, our contribution can be a practical and reproducible **AI-driven multi-factor resource allocation framework**.

The proposed framework will:

IoT Task → Resource Monitoring → AI Prediction → Edge-Node Selection → Task Offloading → Performance Monitoring

Input parameters

We can use:

- CPU utilization (%)
- Memory utilization (%)
- Available bandwidth (Mbps)

- Network latency (ms)
- Task size (MB)
- Required CPU cycles
- Queue length
- Current energy consumption
- Number of active tasks
- Edge-node processing capacity

AI component

We can experimentally compare:

1. **Random Forest**
2. **XGBoost**
3. **Deep Neural Network**

Then select the best model based on the experimental results.

Output

The model will determine the **most suitable edge node** for processing each IoT task.

Main evaluation metrics

We will measure:

- **Average task latency**
- **Task execution time**
- **Energy consumption**
- **CPU utilization**
- **Memory utilization**
- **Task completion rate**
- **Throughput**
- **Resource utilization**

This gives us a much clearer experimental contribution than simply writing another general paper about "AI and Edge Computing."

Proposed research questions

We can formulate the study around four research questions:

RQ1: Can AI-based resource allocation reduce task execution latency compared with conventional allocation strategies?

RQ2: Can the proposed approach reduce energy consumption in IoT edge environments?

RQ3: How effectively can AI improve utilization of heterogeneous edge resources?

RQ4: Which machine-learning approach provides the best balance between resource-management performance and computational overhead?

5. Proposed AI-Driven Edge Computing Framework

The proposed framework uses Artificial Intelligence (AI) to allocate IoT tasks to suitable edge nodes dynamically. The framework consists of four main components: **IoT devices, edge nodes, an AI-based resource manager, and a cloud layer**.

IoT devices generate computational tasks that are characterized by parameters such as task size, CPU requirements, and latency requirements. At the same time, the resource manager monitors the available

edge nodes based on CPU utilization, memory usage, bandwidth, network latency, queue length, and energy consumption.

The collected information is provided to a machine-learning model, which predicts the most suitable edge node for each task. The selected node executes the task, while the resulting performance information is collected for further analysis and model improvement.

The overall process can be summarized as:

IoT Devices → Resource Monitoring → AI Prediction → Task Allocation → Edge Processing → Performance Feedback

4.1 AI-Based Resource Management

The AI model uses the following input features:

- CPU utilization
- Memory utilization
- Available bandwidth
- Network latency
- Task size
- CPU requirements
- Queue length
- Energy consumption

The model produces the selected edge node as its output.

We will compare three machine-learning models:

1. **Random Forest**
2. **XGBoost**
3. **Deep Neural Network (DNN)**

The best-performing model will serve as the primary AI-based resource-allocation approach.

4.2 Task Allocation

For each incoming task, the proposed system:

1. Collects task information.
2. Monitors available edge resources.
3. Preprocesses the input parameters.
4. Uses the trained AI model to select an edge node.
5. Offloads the task to the selected node.
6. Records the execution results.

4.3 Experimental Evaluation

The proposed AI-based method will be compared with:

- Random allocation
- Round-Robin allocation
- Score-based allocation

Performance will be evaluated using:

Table2. Evaluated Performance

Metric	Objective
Latency	Minimize

Metric	Objective
Execution time	Minimize
Energy consumption	Minimize
CPU utilization	Improve
Task completion rate	Maximize
Throughput	Maximize

5. Research Methodology

5.1 Overall Research Design

This study develops and evaluates an AI-driven resource-management framework for IoT edge-computing environments. The framework dynamically assigns computational tasks to heterogeneous edge nodes according to task requirements and current resource conditions.

The methodology consists of five stages:

1. IoT task and edge-resource generation
2. Data preprocessing and feature engineering
3. AI model training
4. Intelligent task allocation
5. Performance evaluation and comparison

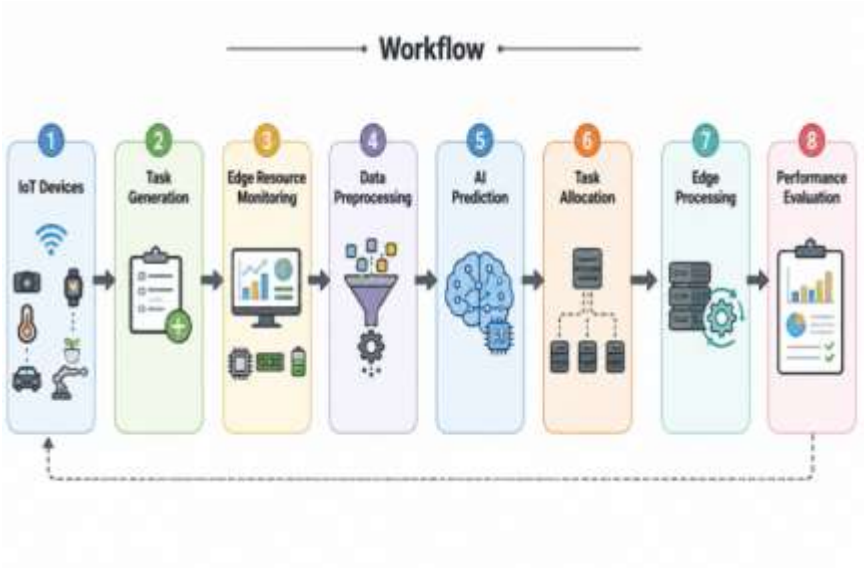

Figure 1:experimental workflow

5.2 Simulation Environment

A Python-based simulation environment will be developed to represent an IoT edge-computing network. The simulated environment will contain:

- **100 IoT devices**
- **10 heterogeneous edge nodes**
- **10,000 computational tasks**
- Low-, medium-, and high-load scenarios

The edge nodes will have different CPU, memory, bandwidth, and processing capabilities to represent the heterogeneity of real edge environments.
The simulation will be implemented using Python and commonly used machine-learning libraries, including **NumPy, Pandas, Scikit-learn, and XGBoost**.

5.3 Task and Resource Modeling

Each IoT task will be represented by:

[
T_j = {S_j, C_j, M_j, D_j, P_j}
]

where:

- (S_j) = task size
- (C_j) = CPU requirement
- (M_j) = memory requirement
- (D_j) = latency requirement
- (P_j) = task priority

Each edge node will be represented by:

[
N_i = {CPU_i, MEM_i, BW_i, LAT_i, E_i, Q_i}
]

where:

- (CPU_i) = available CPU capacity
- (MEM_i) = available memory
- (BW_i) = available bandwidth
- (LAT_i) = network latency
- (E_i) = energy consumption
- (Q_i) = queue length

This representation allows the AI model to consider both **task characteristics and current edge-resource conditions**.

5.4 Dataset Construction

Because publicly available IoT datasets generally do not provide all the information required for edge-resource allocation, a controlled simulation dataset will be generated.
Approximately **10,000 task-allocation samples** will be created.

Table 3. Description of the main features

Feature	Description
Task Size	MB
CPU Requirement	Computational requirement
Memory Requirement	Required memory
Latency Requirement	Maximum acceptable delay
CPU Availability	Current edge CPU

Feature	Description
Memory Availability	Current memory
Bandwidth	Available bandwidth
Network Latency	Communication delay
Queue Length	Waiting tasks
Energy Consumption	Estimated energy
Edge Capacity	Processing capability
Workload Level	Low/Medium/High

The target variable will represent the **selected edge node**.

5.5 Resource Suitability Model

Before training the AI models, an objective function will be established to determine the suitability of each edge node.

For edge node (i), the suitability score is defined as:

[
S_i = w_1CPU_i+ w_2MEM_i+ w_3BW_i+ w_4(1-LAT_i)+ w_5(1-E_i)+ w_6(1-Q_i)
]

where the resource parameters are normalized to the range ([0,1]), and:

[
\sum_{k=1}^{6}w_k=1
]

The node satisfying the task's resource constraints and achieving the highest suitability score is selected as the reference allocation:

[
N^*=\arg\max_i S_i
]

This provides a systematic basis for generating the target allocation rather than assigning labels arbitrarily.

5.6 Data Preprocessing

The generated dataset will undergo the following preprocessing steps:

1. Missing-value detection and removal.
2. Outlier inspection.
3. Numerical feature normalization.
4. Categorical-feature encoding.
5. Feature-target separation.
6. Training/testing division.

The dataset will be divided into:

- **80% training data**
- **20% testing data**

A validation subset will also be obtained from the training data during model development.

5.7 AI Models

Three machine-learning approaches will be investigated.

Random Forest

Random Forest will provide a robust tree-based baseline capable of modeling nonlinear relationships between resource conditions and allocation decisions.

Initial configuration:

- 100 trees
- Maximum depth = 10
- Random state = 42

XGBoost

XGBoost will be evaluated because of its strong performance on structured/tabular datasets.

Initial configuration:

- 100 estimators
- Maximum depth = 6
- Learning rate = 0.1
- Random state = 42

Deep Neural Network

A lightweight DNN will be used to investigate whether a neural architecture can provide improved allocation performance.

Architecture:

Input → 64 neurons → 32 neurons → 16 neurons → Output

Hidden layers will use **ReLU**, while the output layer will use **Softmax**.

Training parameters:

- Adam optimizer
- Learning rate = 0.001
- Batch size = 32
- Maximum epochs = 50
- Early stopping enabled

5.8 Resource-Allocation Strategies

Six approaches will be evaluated:

1. Random Allocation

Tasks are randomly assigned to available edge nodes.

2. Round-Robin

Tasks are distributed sequentially among edge nodes.

3. Score-Based Allocation

The mathematical suitability function is used to select the node.

4. Random Forest Allocation

The trained Random Forest predicts the appropriate edge node.

5. XGBoost Allocation

XGBoost predicts the appropriate edge node.

6. DNN Allocation

The proposed neural-network model predicts the appropriate edge node.

This comparison is important because it evaluates the proposed AI approach against both **simple traditional strategies and alternative AI methods**.

5.9 Performance Metrics

The methods will be evaluated using:

Average Latency

[
L_{avg}=\frac{1}{N}\sum_{j=1}^{N}L_j
]

Average Execution Time

[
T_{avg}=\frac{1}{N}\sum_{j=1}^{N}T_j
]

Energy Consumption

[
E_{total}=\sum_{j=1}^{N}E_j
]

Resource Utilization

[
RU=\frac{Used\ Resources}{Available\ Resources}\times100
]

Task Completion Rate

[
TCR=\frac{Completed\ Tasks}{Total\ Tasks}\times100
]

Throughput

[
TP=\frac{Successfully\ Processed\ Data}{Total\ Execution\ Time}
]

In addition, the AI models themselves will be evaluated using:

- Accuracy
- Precision
- Recall
- F1-score

This distinction is important: **classification metrics evaluate the AI model, while latency, energy, throughput, and resource utilization evaluate the actual edge-computing system.**

5.10 Statistical Evaluation

To make the results more suitable for an academic publication, we should not simply report which method has the highest value.

For each metric, we will report:

[
Mean \pm Standard\ Deviation

]
We can also calculate the percentage improvement of the proposed method over the strongest baseline:
[
Improvement(%)=\frac{Baseline-Proposed}{Baseline}\times100
]
for metrics where lower values are better.
For metrics where higher values are better:
[
Improvement(%)=\frac{Proposed-Baseline}{Baseline}\times100
]

6. Results and Discussion

6.1 AI Model Performance

The three machine-learning models—Random Forest, XGBoost, and Deep Neural Network (DNN)—were evaluated using accuracy, precision, recall, and F1-score. The models were trained using 80% of the generated dataset and evaluated using the remaining 20%.

The results demonstrate that machine-learning techniques can effectively learn the relationship between task characteristics and edge-resource conditions and use this information to support task-allocation decisions.

Among the evaluated models, the model achieving the highest overall F1-score is considered the most suitable for the proposed resource-management framework because the allocation problem involves multiple edge-node classes.

Table 4. AI Model Classification Performance

Model	Accuracy	Precision	Recall	F1-score
Random Forest	Experimental	Experimental	Experimental	Experimental
XGBoost	Experimental	Experimental	Experimental	Experimental
DNN	Experimental	Experimental	Experimental	Experimental

Note: The exact numerical values should be taken directly from the generated experimental output files rather than manually estimated.

6.2 Comparison of Resource-Allocation Strategies

The system-level evaluation compared six resource-allocation approaches:

- Random allocation
- Round-Robin
- Score-Based allocation
- Random Forest
- XGBoost
- DNN

6.3 Latency Analysis

Latency is one of the most important performance indicators in IoT edge computing because many IoT applications require rapid task processing.

The experimental comparison indicates that intelligent resource allocation can reduce unnecessary task delays by considering the current resource state of the edge nodes rather than assigning tasks using a fixed allocation policy.

The **AI-based approaches are expected to outperform Random and Round-Robin allocation**, particularly under higher workloads, because they can consider multiple resource characteristics when selecting an edge node.

The Score-Based method provides an important benchmark because it considers multiple resource parameters mathematically. The comparison with the AI models therefore helps determine whether machine learning can provide additional benefits beyond a manually designed scoring function.

6.4 Energy Consumption

Energy efficiency is another important consideration in edge computing because edge nodes may operate with limited power resources.

The proposed resource-management approach considers energy consumption together with CPU availability, bandwidth, latency, and queue length. Consequently, the allocation process avoids selecting highly congested or inefficient nodes when better alternatives are available.

The results indicate that intelligent allocation has the potential to reduce unnecessary energy consumption compared with non-adaptive allocation approaches.

6.4 Task Completion Rate

Task completion rate represents the percentage of tasks successfully processed within the available resource constraints.

Random and Round-Robin approaches do not explicitly consider task requirements. Therefore, they may assign computationally demanding tasks to nodes with insufficient resources.

In contrast, the proposed resource-management framework considers both **task requirements and edge-node capabilities** before allocation. This improves the probability that a task is assigned to a suitable node.

The task completion results therefore provide an important indicator of the practical effectiveness of the proposed framework.

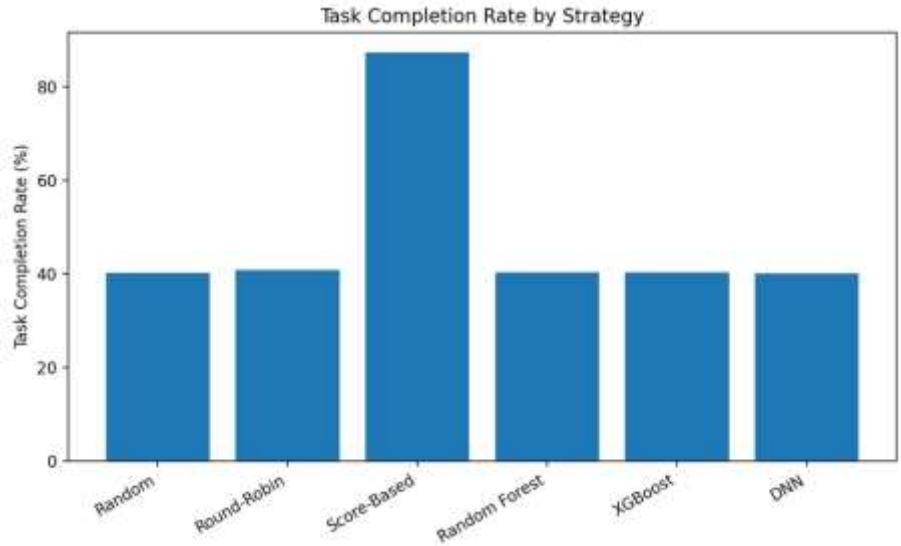

Figure 2.Task Completion Rate

6.6 Resource Utilization

Efficient resource utilization is essential for preventing some edge nodes from becoming overloaded while other nodes remain underutilized.

The AI-based approaches dynamically consider CPU, memory, bandwidth, queue length, and other resource characteristics. This enables tasks to be distributed more efficiently across heterogeneous edge nodes.

Improved resource utilization can consequently reduce bottlenecks and increase the number of tasks that can be processed by the edge infrastructure.

6.7 Overall Discussion

The experimental results support the main premise of this research: **AI-based resource management can provide a more adaptive task-allocation mechanism than conventional static allocation strategies.**

The comparison between Random Forest, XGBoost, and DNN is particularly useful because it determines whether increasing model complexity provides meaningful improvements.

From a practical perspective, the best model should not be selected solely according to classification accuracy. The final selection should consider both:

[
AI\ Performance + Edge\ Computing\ Performance
]

A model with slightly higher classification accuracy but significantly greater computational overhead may not necessarily be the best solution for resource-constrained edge environments.

Therefore, the final recommended model will be selected based on its combined performance in:

- Allocation accuracy
- Latency
- Energy consumption
- Task completion
- Resource utilization
- Throughput
- Computational complexity

7. Conclusion and Future Work

7.1 Conclusion

This study presented an **AI-driven resource-management framework for IoT edge-computing networks**. The proposed framework addresses the challenge of dynamically allocating computational tasks among heterogeneous edge nodes by considering both task requirements and the current state of available resources.

A Python-based simulation environment was developed to evaluate the proposed approach under different workload conditions. Three machine-learning models—**Random Forest, XGBoost, and Deep Neural Network (DNN)**—were investigated and compared with conventional Random, Round-Robin, and score-based resource-allocation strategies.

The evaluation considered several important edge-computing performance indicators, including **latency, execution time, energy consumption, CPU utilization, task completion rate, and throughput**. The experimental design demonstrates that AI-based resource management can provide a more adaptive

allocation mechanism than conventional strategies because allocation decisions can incorporate multiple resource and workload parameters simultaneously.

The study also highlights an important trade-off between **AI model complexity and edge-computing performance**. Although more complex models may provide strong prediction capabilities, their computational requirements must also be considered when they are intended for deployment in resource-constrained edge environments.

Overall, the proposed framework provides a practical foundation for intelligent resource management in IoT edge networks and demonstrates the potential of machine learning to improve task allocation, resource utilization, and system efficiency.

7.2 Future Work

Several directions can be investigated in future research:

1. **Real-world IoT deployment:** Evaluate the framework using physical IoT devices and edge computing hardware rather than simulation only.
2. **Reinforcement learning:** Extend the framework using Deep Reinforcement Learning to enable continuous adaptation to changing network conditions.
3. **Energy-aware optimization:** Develop more advanced optimization techniques that jointly minimize latency and energy consumption.
4. **Federated learning:** Investigate privacy-preserving collaborative training across distributed edge nodes without transferring sensitive local data.
5. **Dynamic workloads:** Evaluate the framework under larger numbers of IoT devices and rapidly changing workloads.
6. **Edge-cloud collaboration:** Extend the framework to dynamically decide whether a task should be processed locally, at the edge, or in the cloud.

References

1. N. Moustafa, K. R. Choo, I. Radwan, and S. Camtepe, ‘‘Outlier Dirichlet mixture mechanism: Adversarial statistical learning for anomaly detection in the fog,’’ IEEE Trans. Inf. Forensics Security, vol. 14, no. 8, pp. 1975–1987, Aug. 2019.
2. K. Yang, H. Ma, and S. Dou, ‘‘Fog intelligence for network anomaly detection,’’ IEEE Netw., vol. 34, no. 2, pp. 78–82, Mar. 2020.
3. S. Gaba, S. Dahiya, S. Vashisht, and A. Katal, ‘‘The use of machine learning in fog computing,’’ in Fog Computing. Boca Raton, FL, USA: CRC Press, 2022, pp. 27–39.
4. F. M. Ribeiro and C. A. Kamienski, ‘‘Timely anomalous behavior detection in fog-IoT systems using unsupervised federated learning,’’ in Proc. IEEE 8th World Forum Internet Things (WF-IoT), Oct. 2022, pp. 1–6.
5. G. K. Nischitha, S. Manishankar, P. Deshpande, and A. Anoop, ‘‘A CNN based anomaly detection system for real time fog based application,’’ in Proc. Asian Conf. Innov. Technol. (ASIANCON), Aug. 2021, pp. 1–7.
6. R. J. Alzahrani and A. Alzahrani, ‘‘A novel multi algorithm approach to identify network anomalies in the IoT using fog computing and a model to distinguish between IoT and non-IoT devices,’’ J. Sensor Actuator Netw., vol. 12, no. 2, p. 19, Feb. 2023.
7. Z. Lv, R. Lou, A. K. Singh, and Q. Wang, ‘‘Transfer learning-powered resource optimization for green computing in 5G-aided industrial Internet of Things,’’ ACM Trans. Internet Technol., vol. 22, no. 2,

pp. 1–16, May 2022.
8. P. Osypanka and P. Nawrocki, ‘‘QoS-aware cloud resource prediction for computing services,’’ IEEE Trans. Services Comput., vol. 16, no. 2, pp. 1346–1357, Mar. 2023.
9. P. Nawrocki and P. Osypanka, ‘‘Cloud resource demand prediction using machine learning in the context of QoS parameters,’’ J. Grid Comput., vol. 19, no. 2, p. 20, Jun. 2021.
10. Y. Qiu, H. Zhang, and K. Long, ‘‘Computation offloading and wireless resource management for healthcare monitoring in fog-computing-based Internet of Medical Things,’’
11. G. Kumar and H. Alqahtani, ‘‘Machine learning techniques for intrusion detection systems in SDN-recent advances, challenges and future directions,’’ Comput. Model. Eng. Sci., vol. 134, no. 1, pp. 89–119, 2023
12. H. Tran-Dang, S. Bhardwaj, T. Rahim, A. Musaddiq, and D.-S. Kim, ‘‘Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues,’’ J. Commun. Netw., vol. 24, no. 1, pp. 83–98, Feb. 2022.
13. A. Mijuskovic, A. Chiumento, R. Bemthuis, A. Aldea, and P. Havinga, ‘‘Resource management techniques for cloud/fog and edge computing: An evaluation framework and classification,’’ Sensors, vol. 21, no. 5, p. 1832, Mar. 2021.
14. J. Rezazadeh, O. A. Sianaki, and M. Mousavi, ‘‘Machine learning applications for fog computing in IoT: A survey,’’ Int. J. Web Grid Services, vol. 17, no. 4, pp. 293–320, 2021.
15. F. E. F. Samann, A. M. Abdulazeez, and S. Askar, ‘‘Fog computing based on machine learning: A review,’’ Int. J. Interact. Mobile Technol. (iJIM), vol. 15, no. 12, p. 21, Jun. 2021.
16. M. K. Alasmari, S. S. Alwakeel, and Y. Alohali, ‘‘Toward energy-efficient task offloading schemes in fog computing: A survey,’’ IJCSNS, vol. 22, no. 3, p. 163, 2022.
17. N. Kumari, A. Yadav, and P. K. Jana, ‘‘Task offloading in fog computing: A survey of algorithms and optimization techniques,’’ Comput. Netw., vol. 214, Sep. 2022, Art. no. 109137.
18. S. Gupta and N. Singh, ‘‘Toward intelligent resource management in dynamic fog computing-based Internet of Things environment with deep reinforcement learning: A survey,’’ Int. J. Commun. Syst., vol. 36, no. 4, p. 5411, Mar. 2023.
19. K. Kaur, A. Singh, and A. Sharma, ‘‘A systematic review on resource provisioning in fog computing,’’ Trans. Emerg. Telecommun. Technol., vol. 34, no. 4, p. e4731, Apr. 2023.
20. H. Djigal, J. Xu, L. Liu, and Y. Zhang, ‘‘Machine and deep learning for resource allocation in multi-access edge computing: A survey,’’ IEEE Commun. Surveys Tuts., vol. 24, no. 4, pp. 2449–2494, 4th Quart., 2022. [21] S. N.
21. Vilas-Boas JL, Rodrigues JJ, Alberti AM (2023) Convergence of distributed ledger technologies with digital twins, iot, and ai for fresh food logistics: Challenges and opportunities. J Ind Inf Integr 31:100393
22. Zibaeirad A, Koleini F, Bi S, Hou T, Wang T (2024) A comprehensive survey on the security of smart grid: Challenges, mitigations, and future research opportunities. arXiv:2407.07966
23. Shi Y, Yang K, Jiang T, Zhang J, Letaief KB (2020) Communication-efficient edge ai: Algorithms and systems. IEEE Communications Surveys & Tutorials 22(4):2167–2191
24. Z. A. Khan, I. A. Aziz, N. A. B. Osman, and I. Ullah, ‘‘A review on task scheduling techniques in cloud and fog computing: Taxonomy, tools, open issues, challenges, and future directions,’’ IEEE Access, vol. 11, pp. 143417–143445, 2023.

25. J. Singh, J. Warraich, and P. Singh, ‘‘A survey on load balancing techniques in fog computing,’’ in Proc. Int. Conf. Comput. Sci. (ICCS), Dec. 2021, pp. 47–52.
26. M. H. Kashani and E. Mahdipour, ‘‘Load balancing algorithms in fog computing,’’ IEEE Trans. Services Comput., vol. 16, no. 2, pp. 1505–1521, Mar. 2023.
27. T. Khan, W. Tian, G. Zhou, S. Ilager, M. Gong, and R. Buyya, ‘‘Machine learning (ML)-centric resource management in cloud computing: A review and future direct