iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VII **Month of publication:** July 2026

DOI: https://doi.org/10.22214/ijraset.2026.84091

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

Adaptive Cloud-Edge Scheduler Using Lightweight AI Models for Real-Time IoT Streams

Munesula Venkatesh[1], Dr. M. Saravanan[2]
[1, 2]*Department of Computer Science and Engineering Holy Mary Institute of Technology & Science Hyderabad, Telangana, India*

Abstract: ***The rapid growth of Internet of Things (IoT) devices has increased the volume of real-time data generated by sensors, smart devices, industrial controllers, and healthcare monitoring systems. Cloud-centric processing provides scalable computation, but it often introduces high communication latency, bandwidth overhead, and energy cost for delay-sensitive IoT streams. Edge and fog computing reduce response time by moving computa-tion closer to data sources, but efficient workload placement across edge, fog, and cloud resources remains a challenging scheduling problem. This paper proposes an adaptive cloud-edge scheduler using lightweight artificial intelligence models for real-time IoT stream placement. The framework dynamically selects edge, fog, or cloud execution based on stream priority, latency deadline, bandwidth availability, workload complexity, CPU uti-lization, memory utilization, queue length, energy availability, and historical scheduling success. Lightweight models such as Decision Tree, Logistic Regression, Random Forest, and Tiny Neural Network are considered to support fast inference under resource constraints. A latency-energy-aware cost function vali-dates model predictions and improves runtime decision quality. The framework also integrates explainable scheduling decisions and a human-in-the-loop override mechanism for operational governance. Experimental evaluation using a working prototype demonstrates adaptive distribution of IoT streams across edge, fog, and cloud layers, with 11 observed scheduling decisions, 55% success rate, 274.796 ms average latency, and 0.111 J average energy consumption. The results indicate that combining lightweight AI prediction with cost-aware validation improves scheduling flexibility, transparency, and practical applicability in real-time IoT systems.***

Index Terms: ***Cloud-edge computing, edge computing, fog computing, Internet of Things, real-time stream processing, lightweight machine learning, adaptive scheduling, latency op-timization, energy efficiency, explainable AI.***

I. INTRODUCTION

The Internet of Things (IoT) has become a key technology for smart healthcare, industrial automation, smart cities, in-telligent transportation, environmental monitoring, and energy management. IoT systems continuously generate data streams from distributed sensors and embedded devices. Many of these streams require real-time or near real-time processing because delayed decisions can reduce quality of service, safety, or operational efficiency. Traditional cloud computing provides elastic storage and high-performance analytics, but routing all raw IoT data to remote cloud servers increases communication latency, bandwidth consumption, and energy overhead. Edge computing and fog computing address this limitation by placing computation close to data sources. Edge nodes can process urgent streams locally, while fog nodes provide intermediate regional processing and cloud servers provide large-scale analytics. However, deciding where each stream should be processed is difficult because IoT workloads are heterogeneous and infrastructure conditions change dynami-cally. Edge nodes may have strict energy and memory limits, fog nodes may become congested, and cloud servers may in-troduce high network delay. Existing surveys on edge and fog scheduling identify resource heterogeneity, workload mobility, latency constraints, and energy efficiency as major challenges [1]–[3].

Static scheduling policies are insufficient for such dynamic environments. A fixed threshold policy may route streams to the edge when latency is strict, but this can overload the edge node when CPU utilization or queue length is high. Simi-larly, cloud-only scheduling can support high computation but may violate latency deadlines. Recent research has explored machine learning, deep learning, and reinforcement learning for computation offloading and resource allocation [4]–[6]. However, complex models may introduce high inference cost and may not be suitable for lightweight edge deployment.

This paper proposes an adaptive cloud-edge scheduler that uses lightweight AI models and a latency-energy cost function to select the best processing layer for each IoT stream. The system also includes explainability and human-in-the-loop (HITL) support to improve trust and governance.

The major contributions of this paper are:

1) An adaptive cloud-edge scheduling framework for real-time IoT streams.
2) A lightweight AI-based decision model for selecting edge, fog, or cloud processing.
3) A latency-energy-aware cost function for runtime decision validation.
4) A stream routing mechanism with failover support.
5) Explainable scheduling decisions with HITL override capability.
6) Experimental evaluation using latency, energy, confidence, success rate, and layer distribution metrics.

II. RELATED WORK

Cloud, fog, and edge computing have been widely studied as complementary paradigms for distributed IoT processing. Edge computing brings computation close to data sources and reduces response delay, while fog computing provides intermediate aggregation and coordination. Cloud computing remains useful for compute-intensive and delay-tolerant ana-lytics. Luo et al. reviewed resource scheduling techniques in edge computing and highlighted the importance of resource heterogeneity, mobility, and latency-aware scheduling [1]. Goudarzi et al. surveyed IoT application scheduling in edge and fog environments and identified latency, energy, reliability, and application constraints as key scheduling dimensions [2]. Jamil et al. discussed task scheduling in fog and Internet of Everything systems, emphasizing the need for efficient resource allocation [3].

IoT stream processing requires fast decision-making un-der continuously changing conditions. Resource-efficient dis-tributed AI has become important for IoT applications because transmitting all data to the cloud is often inefficient [7], [8]. Murshed et al. reviewed machine learning at the network edge and emphasized the need for models that can operate under resource constraints [9]. Singh and Gill discussed Edge AI and highlighted its role in latency reduction, privacy improvement, and autonomous decision-making [10].

Machine learning and deep learning have been applied to resource allocation and computation offloading. Djigal et al. surveyed machine and deep learning methods for resource allocation in multi-access edge computing [4]. Kar et al. reviewed offloading methods across cloud, edge, and fog systems and reported that both optimization and learning-based approaches are widely used [5]. Deep reinforcement learning has also been investigated for adaptive scheduling [6], [11], but such approaches may require extensive training and may be too heavy for low-power devices.

Trust and explainability are also important for AI-enabled scheduling. Wang et al. described trustworthy edge intelligence and identified transparency, reliability, security, and sustain-ability as major requirements [12]. Vouros discussed explain-able deep reinforcement learning and the need to interpret complex AI decisions [13]. Existing approaches often focus on scheduling performance but provide limited support for explainability and HITL governance. This paper addresses this gap by integrating lightweight AI, cost-aware validation, explainable decision traces, and human override.

III. PROPOSED SYSTEM ARCHITECTURE

The proposed system consists of six major layers: IoT stream generation, edge processing, fog coordination, cloud analytics, AI scheduling, and governance. The IoT stream layer generates real-time sensor data with metadata such as priority, payload size, latency deadline, and workload complexity. The edge layer performs local processing and monitors resource status. The fog layer provides intermediate processing and regional coordination. The cloud layer provides high-compute analytics and long-term storage. The AI schedul-

TABLE I
SUMMARY OF RELATED WORK

Area	Focus	Limitation
Edge scheduling [1]	Resource allocation and edge constraints	Limited HITL support
Fog scheduling [3]	Task placement in fog/IoE	Limited explainability
Cloud-edge offloading [5]	Offloading across continuum	Heavy models common
Edge AI [10]	Local intelligence	Deployment constraints
DRL scheduling [11]	Adaptive resource control	High training overhead
Trustworthy edge intelligence [12]	Transparency and reliability	Needs domain integration

ing layer predicts the processing location using lightweight models. The governance layer provides explainability, HITL override, logging, and feedback. Fig. 1 shows the proposed architecture.

IV. METHODOLOGY

The scheduling problem is formulated as a multi-class decision problem. For each incoming stream, the scheduler selects one processing layer from

$$K = \{Edge, Fog, Cloud\}. \quad (1)$$

The input feature vector is defined as

$$\mathbf{x} = [A, P, W, D, B, N, C, M, Q, E, H] \quad (2)$$

where A is arrival rate, P is stream priority, W is workload size, D is latency deadline, B is bandwidth availability, N is network delay, C is CPU utilization, M is memory utilization, Q is queue length, E is energy availability, and H is historical success rate. Table II describes the feature vector.

TABLE II
FEATURE VECTOR DESCRIPTION

Symbol	Description	Type
A	Stream arrival rate	Stream
P	Priority score	Stream
W	Workload complexity	Stream
D	Latency deadline	QoS
B	Available bandwidth	Network
N	Network delay	Network
C	CPU utilization	Resource
M	Memory utilization	Resource
Q	Queue length	Resource
E	Energy availability	Resource
H	Historical success rate	Feedback

The scheduling function is represented as

$$d = f(\mathbf{x}), d \in K. \quad (3)$$

The latency model is given by

$$L_{total} = L_{tx} + L_q + L_p, \quad (4)$$

where

$$L_{tx} = \frac{S}{R} + N, \quad L_q = \frac{Q}{\mu}, \quad L_p = \frac{W}{F}. \quad (5)$$

Here, S is stream size, μ is service rate, and F is processing capacity. Energy is modeled as

$$E_{total} = E_{tx} + E_p, \quad (6)$$

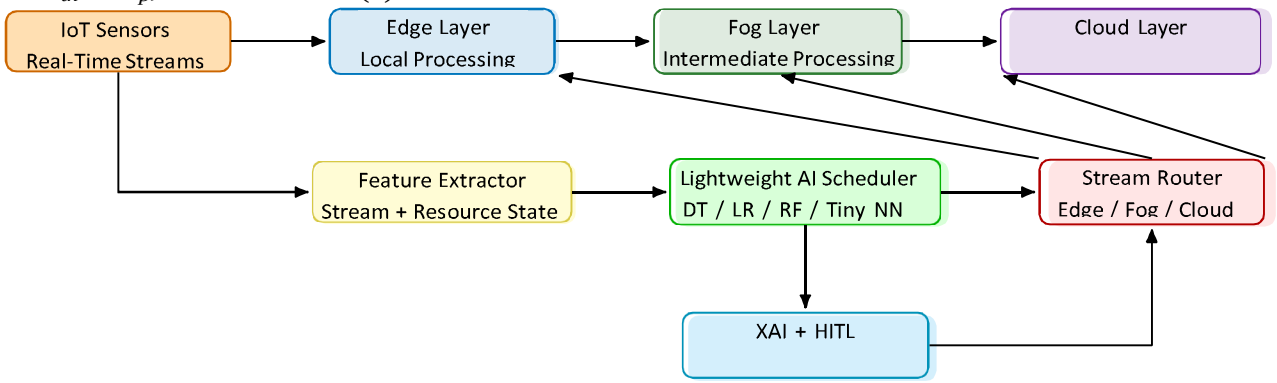


Fig. 1. Architecture of the proposed adaptive cloud-edge scheduling framework.

where

$$E_{tx} = P_{tx} \times L_{tx}, E_p = P_{cpu} \times L_p. \quad (7)$$

The multi-objective cost is defined as

$$C = \alpha L + \beta E + \gamma U + \delta V, \quad (8)$$

where L is normalized latency, E is normalized energy, U is resource utilization penalty, and V is QoS violation penalty. Fig. 2 shows the scheduling decision workflow.

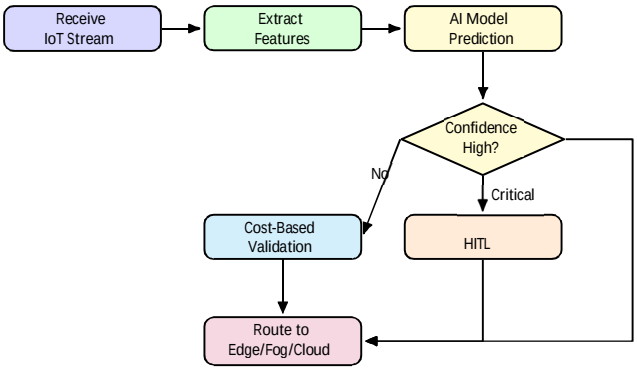


Fig. 2. Scheduling decision workflow.

V. IMPLEMENTATION

The prototype was implemented as a local software system using Python, FastAPI, SQLite, SQLAlchemy, Scikit-learn, Pandas, NumPy, Joblib, HTML, CSS, JavaScript, Bootstrap, and Chart.js. The implementation includes IoT stream simu-lators, resource monitors, a feature extractor, lightweight AI models, a latency-energy model, a decision policy module, a stream router, explainability logic, HITL override support, and result logging.

The stream simulator generates data for healthcare, traf-fic, industrial, and environmental sensors. Resource monitors simulate CPU utilization, memory utilization, queue length, bandwidth, energy level, and network delay for edge, fog, and cloud layers. The AI scheduler loads trained models from local storage and returns a predicted layer and confidence score. The decision policy validates this prediction using runtime

Algorithm 1 Adaptive Cloud-Edge Scheduling Algorithm

Require: IoT stream s_i, resource states R, trained model f, threshold ϑ
Ensure: Final processing layer d_i^{final}

```
Extract stream metadata and collect edge/fog/cloud states
Form normalized feature vector x_i
Predict layer d^_i = f(x_i) and confidence conf_i
for each layer k ∈ {Edge, Fog, Cloud} do
    Estimate latency L(s_i, k) and energy E(s_i, k)
    Compute cost C(s_i, k)
end for
if conf_i ≥ ϑ and constraints are satisfied then
    d_i^final ← d^_i
else if stream is critical then
    Request HITL review and set d_i^final from human decision
else
    d_i^final ← arg min_k C(s_i, k)
end if
if selected layer is unavailable then
    Apply failover to next available layer
end if
Route stream and store feedback
```

constraints and cost estimates. The dashboard presents total decisions, success rate, average latency, average energy, layer utilization, decisions by layer, latency-energy charts, decision logs, and HITL overrides.

The system can be executed locally using the following workflow: generate dataset, train models, run FastAPI back-end, open the dashboard, execute demo scheduling requests, and export result logs. No paid cloud service or external API dependency is required.

VI. RESULTS AND DISCUSSION

The prototype was evaluated using a demonstration run of 11 scheduling decisions. The dashboard reported a success rate of 55%, average latency of 274.796 ms, and average energy consumption of 0.111 J. The final layer distribution consisted of 2 edge decisions, 4 fog decisions, and 5 cloud decisions.

This distribution indicates that the scheduler dynamically selected different layers instead of using a fixed placement strategy.

Table III presents sample decision results.

TABLE III
SCHEDULING DECISION RESULTS

Decision	Stream	Pred.	Final	Conf.	Status
DEC-6553	STR-1E90	FOG	CLOUD	89%	Degraded
DEC-0A77	STR-4CCD	CLOUD	EDGE	100%	Success
DEC-76F7	STR-DAD2	FOG	FOG	90%	Success
DEC-CFE2	STR-0BDB	CLOUD	CLOUD	100%	Success
DEC-48AB	STR-D296	CLOUD	FOG	100%	Degraded

The lowest observed latency in the demonstration was 145.937 ms for fog decisions, while edge decisions showed low energy values such as 0.061 J. Cloud decisions showed higher latency values, such as approximately 350 ms, but remained useful for high-workload or constraint-violating sit-uations. These observations support the need for adaptive scheduling rather than cloud-only or edge-only strategies.

TABLE IV
OBSERVED RESULT SUMMARY

Metric	Value
Total scheduling decisions	11
Successful decisions	6
Degraded decisions	5
Success rate	55%
Average latency	274.796 ms
Average energy	0.111 J
EDGE final decisions	2
FOG final decisions	4
CLOUD final decisions	5

Table V compares the proposed scheduler with baseline methods.

TABLE V
COMPARISON WITH BASELINE SCHEDULING METHODS

Method	Latency	Energy	Adaptivity
Cloud-only	High	Moderate	Low
Edge-only	Low if free	Low	Low
Rule-based	Moderate	Moderate	Limited
Proposed scheduler	Adaptive	Adaptive	High

The proposed scheduler provides better adaptability because it combines AI prediction with runtime cost validation. Fog processing provides a balance between latency and capacity. Edge processing reduces energy and transmission cost when resources are available. Cloud processing supports high work-load tasks or cases where edge and fog constraints are violated. Explainability improves trust by reporting confidence, latency, energy, and decision rationale. HITL support improves gover-nance by allowing administrators to override critical decisions.

VII. CONCLUSION AND FUTURE WORK

This paper presented an adaptive cloud-edge scheduler using lightweight AI models for real-time IoT stream processing. The proposed framework dynamically selects edge, fog, or cloud processing based on stream features, resource con-ditions, latency constraints, bandwidth, energy, and model confidence. A latency-energy-aware cost function validates AI predictions, while explainability and HITL support improve transparency and control.

The prototype results show that the scheduler can distribute streams across edge, fog, and cloud layers and provide de-cision explanations. The observed 55% success rate, 274.796 ms average latency, and 0.111 J average energy demonstrate the feasibility of the prototype under simulated dynamic con-ditions. The framework is suitable for academic demonstration and can be extended for real IoT deployments.

Future work will focus on reinforcement learning-based scheduling, federated learning across edge nodes, Kubernetes-based deployment, digital twin-based scheduling, blockchain-based audit logging, 6G-enabled IoT scheduling, and advanced explainable AI mechanisms.

REFERENCES

[1] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," IEEE Communications Surveys & Tutorials, vol. 23, no. 4, pp. 2131–2165, 2021. [Online]. Available: https://doi.org/10.1109/COMST.2021.3106401

[2] M. Goudarzi, M. Palaniswami, and R. Buyya, "Scheduling iot applications in edge and fog computing environments: A taxonomy and future directions," ACM Computing Surveys, vol. 55, no. 7, pp. 1–41, 2022. [Online]. Available: https://doi.org/10.1145/3544836

[3] B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, "Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions," ACM Computing Surveys, vol. 54, pp. 1–38, 2022. [Online]. Available: https://doi.org/10.1145/3513002

[4] H. Djigal, J. Xu, L. Liu, and Y. Zhang, "Machine and deep learning for resource allocation in multi-access edge computing: A survey," IEEE Communications Surveys & Tutorials, vol. 24, no. 4, pp. 2449–2494, 2022. [Online]. Available: https://doi.org/10.1109/COMST.2022.3199544

[5] B. Kar, W. Yahya, Y. Lin, and A. Ali, "Offloading using traditional optimization and machine learning in federated cloud-edge-fog systems: A survey," IEEE Communications Surveys & Tutorials, vol. 25, no. 2, pp. 1199–1226, 2023. [Online]. Available: https://doi.org/10.1109/COMST.2023.3239579

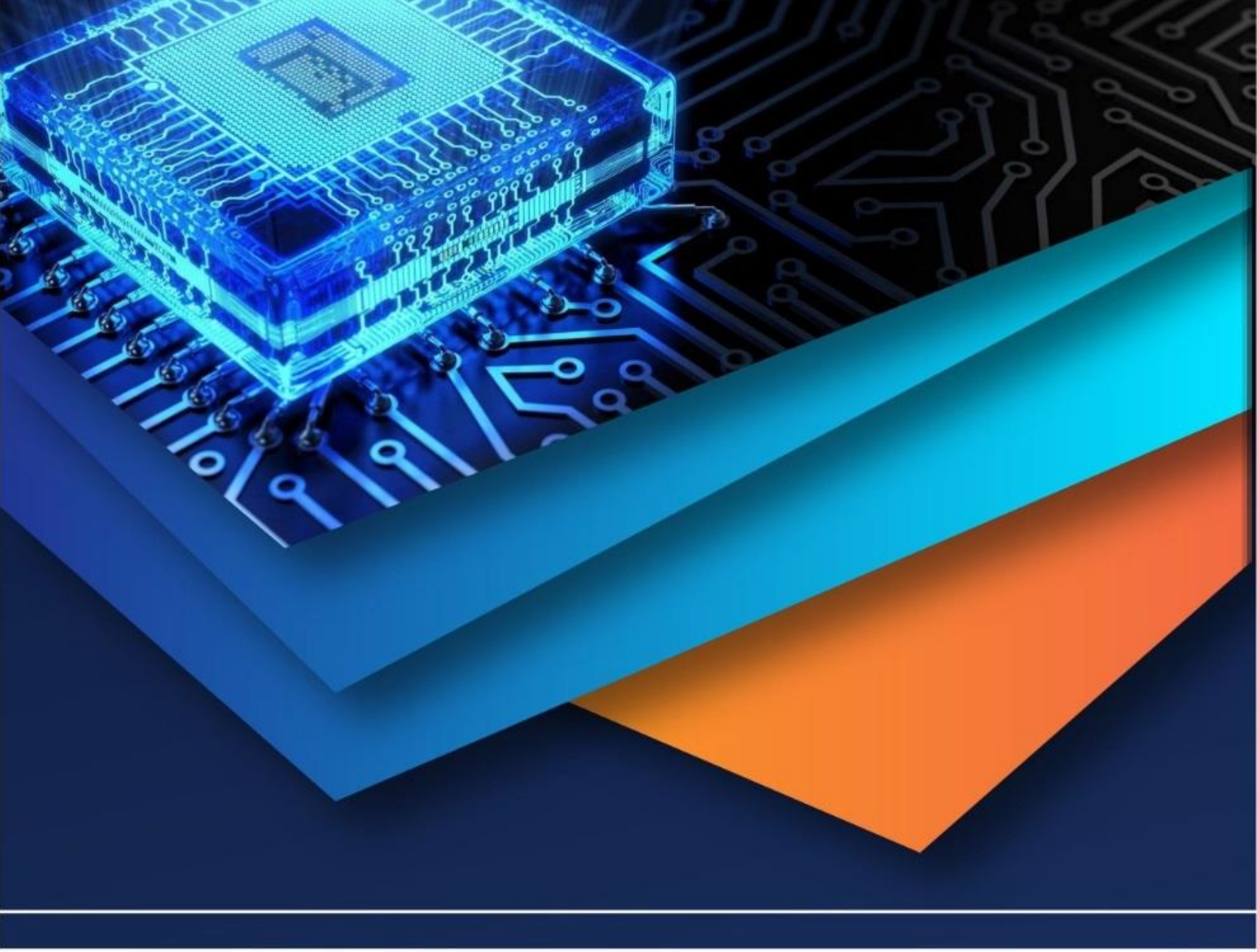
[6] P. Peng, W. Lin, W. Wu, H. Zhang, S. Peng, Q. Wu, and K. Li, "A survey on computation offloading in edge systems: From the perspective of deep reinforcement learning approaches," Computer Science Review, vol. 53, p. 100656, 2024. [Online]. Available: https://doi.org/10.1016/j.cosrev.2024.100656

[7] E. Baccour, N. Mhaisen, A. A. Abdellatif, A. Erbad, A. Mohamed, M. Hamdi, and M. Guizani, "Pervasive ai for iot applications: A survey on resource-efficient distributed artificial intelligence," IEEE Communications Surveys & Tutorials, vol. 24, no. 4, pp. 2366–2418, 2022. [Online]. Available: https://doi.org/10.1109/COMST.2022.3200740

[8] O. Aouedi, T.-H. Vu, A. Sacco, D. C. Nguyen, K. Piamrat, G. Marchetto, and Q. Pham, "A survey on intelligent internet of things: Applications, security, privacy, and future directions," IEEE Communications Surveys & Tutorials, vol. 27, no. 2, pp. 1238–1292, 2024. [Online]. Available: https://doi.org/10.1109/COMST.2024.3430368

[9] M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," ACM Computing Surveys, vol. 54, no. 8, pp. 1–37, 2021. [Online]. Available: https://doi.org/10.1145/3469029

[10] R. Singh and S. S. Gill, "Edge ai: A survey," Internet of Things and Cyber-Physical Systems, vol. 3, pp. 71–92, 2023. [Online]. Available: https://doi.org/10.1016/j.iotcps.2023.02.004



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY