S.A. Jilani et al., Vol 3(1), pp 01-13
https://doi.org/10.55306/CJDTES.2026.030101
e-ISSN: 3048-8389
Ci-STEM Journal of Digital Technologies and Expert Systems

Smart IoT Devices Using AI-Based Energy-Efficient Edge Computing Architecture

Sayamuddin Ahmed Jilani[1], Soumitra Kumar Mandal[2]
Department of Computer Science & Engineering, Maulana Abul Kalam Azad University of Technology, Kolkata, West Bengal, India
[2]Department of Electrical Engineering, National Institute of Technical Teachers' Training & Research (NITTTR), Kolkata, West Bengal, India.
[1]email: 1075sam@gmail.com, [2]email: skmandal@nitttrkol.ac.in.

Open Access
Research Article
Received: 20/03/2026
Accepted: 24/04/2026
Published: 08/05/2026
Corresponding author email:
1075sam@gmail.com
Citation:
S.A. Jilani et al., "Smart IoT Devices Using AI-Based Energy-Efficient Edge Computing Architecture" Ci-STEM Journal of Digital Technologies and Expert Systems, Vol. 3(1), pp. 01-13, 2026
doi: 10.55306/CJDTES.2026.030101
Copyright:
©2026 S.A. Jilani et al.,
This is an open-access article distributed under the terms of the Creative Commons Attribution License which grants the right to use, distribute, and reproduce the material in any medium, provided that proper attribution is given to the original author and source, in accordance with the terms outlined by the license.
(https://creativecommons.org/licenses/by/4.0/)
Published By:
Ci-STEM Global Services Foundation, India.

Abstract:
As the number of IoT-connected smart objects, including environmental, wearable, smart-home, and industrial sensors, keeps growing, delivering low-latency, privacy-preserving intelligence at the network edge is no longer just convenient; it is becoming a requirement. Conventional edge deployments typically rely on fixed resource allocations that pay little attention to fluctuating workloads, device heterogeneity, or battery constraints, so energy ends up being spent needlessly. This paper introduces an edge computing framework built around four coordinated, energy-aware components: a reinforcement-learning (RL) task-offload scheduler, an adaptive dynamic voltage and frequency scaling (DVFS) controller, a dynamic model quantization/pruning engine, and a duty-cycled sensor gating manager. These sit within a four-tier architecture spanning the cloud, edge orchestration, edge compute, and device layers, extending the three-tier structure common in industry practice by introducing an explicit orchestration tier with clearly defined responsibilities. We describe the resulting system architecture and interlayer coordination protocol and evaluate the approach through a discrete-event simulation calibrated against representative embedded-inference power profiles. The simulated results show a 77.6% reduction in energy per inference relative to cloud-only offloading, a 57.0% reduction relative to a statically configured edge baseline, and a 3.7x improvement in projected device battery life, with end-to-end latency scaling sub-linearly as concurrent device load increases. An ablation study shows removing the RL-based offload scheduler causes the largest single drop in performance, with dynamic quantization contributing the next-largest share. We close by addressing real-world deployment challenges, particularly scale and security, and point to federated edge learning and standardized, cross-layer energy orchestration as key directions for future work.
Keywords:
Edge Computing, Internet of Things, Energy Efficiency, Reinforcement Learning, TinyML, Dynamic Voltage and Frequency Scaling, Model Quantization, Task Offloading..

1. INTRODUCTION

Connected IoT devices around the world numbered approximately 21 billion at the end of 2025, with industry predictions of close to 39 billion by 2030[1]. These devices are everywhere: watching over crops, managing smart-city infrastructure, running factory floors, and tracking vitals on wearable trackers[2]. Increasingly, they do much more than collect raw data and transmit it to the cloud. They analyze it in real time, tag anomalies and categorize events, and respond.

This transition is driven by three realities:

- Latency: In many mission critical systems, the latency of a round trip to a remote data center is unacceptable[3].
- Privacy: With increasing regulation and user expectations, it's easier to keep sensitive user data on-device[4].
- Connectivity: Field deployments still more often than not have to contend with unstable, low speed networks[5].

Moving intelligence to the edge resolves bandwidth and latency issues but power is a hard physical limit. Most IoT devices operate on tight milliwatt budgets, powered by small batteries or energy harvesters, and even a modest neural network can drain an unoptimized device in a single day[6]. Today, most deployments are still driven by brittle static rules, e.g. fixed sampling rates, static model architectures, and hard-coded offloading logic which break down as soon as network traffic spikes or a battery starts to fail[7].

In this paper, we consider energy management as a learned continuous optimization problem, rather than a one-time setup task. We propose an AI-driven energy-aware edge architecture that dynamically orchestrates resources between devices and the cloud[8]. An orchestration layer fundamentally manages four active levers:

- Smart Offloading: A reinforcement-learning (RL)-based scheduler makes an on-the-fly decision for each task, choosing between local execution, offloading to a neighboring edge gateway, or offloading to the cloud.
- Dynamic Voltage and Frequency Scaling (DVFS): Chip clock speeds are scaled up or down to match current workload demands[9].
- Adaptive Model Compression: The system dynamically switches between model variants, using pruning or quantization techniques common in TinyML deployments[10], trading fractions of a percentage point in accuracy for large battery savings.
- Context-Aware Duty Cycling: Sensors run at a low duty cycle during quiet periods and ramp up only when meaningful activity is detected[11].

Major Contributions

- Unified Resource Orchestration. We unify offloading, voltage scaling, model compression, and sensor cycling into a single closed-loop control, rather than treating them as separate knobs.
- Telemetry-Driven RL Scheduler. We formulate task execution as a constrained Markov Decision Process (MDP) which enables the agent to constantly trade-off energy, latency and precision in live field conditions.
- Realistic Hardware-Calibrated Validation: We evaluate the system against pure-cloud and static-edge baselines using hardware-calibrated power profiles across lifetime, latency, and inference fidelity.
- Deployment & Ablation Analysis: We isolate the effect of each control lever and analyze the resiliency of the architecture to real world constraints of scaling and security.

Paper roadmap: Section 2 provides a literature review on edge intelligence. The proposed system architecture is introduced in Section 3. The MDP formulation and compression logic are presented in Section 4. Section 5 describes the experimental testbed. Section 6 contains results and discussion. Section 7 covers security and scaling, and Section 8 closes with conclusions and future directions.

2. RELATED WORK

A. Edge and Fog Computing for IoT

Edge and fog architectures distribute computing tasks between local devices and the central cloud by placing computing nodes in between, which lowers latency and saves network bandwidth[12]. A recent survey of edge-cloud collaborative computing finds that most existing work still focuses primarily on where to place tasks and how to balance loads against latency deadlines, treating energy consumption as a secondary concern at best[13]. This motivates architectures that place power efficiency front and center, not as a happy by-product of low-latency scheduling, but as a primary objective to continuously optimize.

B. TinyML and On-Device Inference

TinyML[14] is a movement that has developed a range of techniques for running neural networks on microcontroller-class hardware with only kilobytes of memory to work with[15]. Recent surveys catalog these methods in detail: quantization (reducing numerical precision from 32-bit floating point to 8-bit integer or lower), structured and unstructured pruning (removing redundant weights or channels), and knowledge distillation (training compact student models to approximate larger teacher models)[16]. Even work aimed at compressing large language models for edge deployment follows

the same basic pattern, trading a small accuracy cost for large gains in memory and energy footprint[17]. These techniques reduce both memory footprint and per-inference energy, but they are typically applied once, statically, at deployment time. Some recent efforts have begun exploring joint, multi-objective search over compression and scheduling parameters together[18], but runtime adaptivity to changing battery and workload conditions remains uncommon. The architecture proposed here instead treats compression level as a dynamic control variable that can be adjusted at runtime in response to battery state and workload criticality.

C. Reinforcement Learning for Resource Management

Reinforcement learning has been applied extensively to task offloading and resource allocation in mobile edge computing, where an agent learns a policy that maps system state to offload decisions in order to minimize a weighted cost of latency and energy[19]. A recent survey of deep reinforcement learning approaches to energy-efficient offloading finds that most formulations still optimize offloading in isolation, without coordinating with model compression or sensor duty-cycling[20]. A broader survey of DRL applications across MEC-enabled terrestrial wireless networks[21] reaches a similar conclusion and flags cross-layer coordination between offloading and other resource knobs as an open research direction, multi-agent DRL formulations applied to related resource-allocation problems in vehicular networks show that jointly coordinating several decision variables at once can outperform single-lever optimization[22]. The architecture proposed here treats offloading, compression, and duty-cycling as jointly optimized control levers within a single decision loop, an approach that we argue captures cross-component energy interactions that isolated, single-lever optimization tends to miss.

D. Dynamic Voltage and Frequency Scaling

DVFS is a well-known technique in embedded and mobile systems to trade-off computational throughput against power draw by adjusting processor voltage and clock frequency. Traditional DVFS governors are based on heuristic rules reacting to instantaneous CPU utilization. Even recent surveys on edge-oriented power management note that this reactive design is still the most common industrial practice[23]. Instead of treating DVFS operating-point selection as a separate problem, this work utilizes the same learned scheduling policy that governs task placement. This allows power-scaling decisions to predict future workload instead of merely reacting to it post hoc.

E. Positioning of This Work

Table I compares the proposed architecture with previous approaches. The key difference is the joint, learned coordination of four control levers, task offloading, DVFS, model compression, and sensor duty-cycling, under a common energy-latency-accuracy objective, rather than optimizing any single lever independently[24], [25] .

Table 1: Relative Positioning of the Proposed Architecture to Previous Approaches

Approach Category	Optimization Target	Adaptivity	Cross-Layer Coordination
Static edge/fog placement	Latency, bandwidth	None (fixed rules)	No
Static TinyML compression	Memory, per-inference energy	Deployment-time only	No
RL-based offloading (single-lever)	Latency + energy cost	Runtime, single lever	Partial
Heuristic DVFS governors	Power vs. throughput	Runtime, reactive	No
Proposed AI-driven architecture	Energy, latency, accuracy (joint)	Runtime, predictive, multi-lever	Yes

3. PROPOSED SYSTEM ARCHITECTURE

The proposed architecture comprises four layers, shown in Fig. 1: the Smart IoT Device Layer, the Edge Compute Node Layer, the AI-Driven Edge Orchestration Layer, and the Cloud/Backend Layer.

Control and inference decisions flow downstream from the cloud toward the device, while telemetry and energy feedback flow upstream, closing the loop that lets the orchestration layer continually revise its decisions

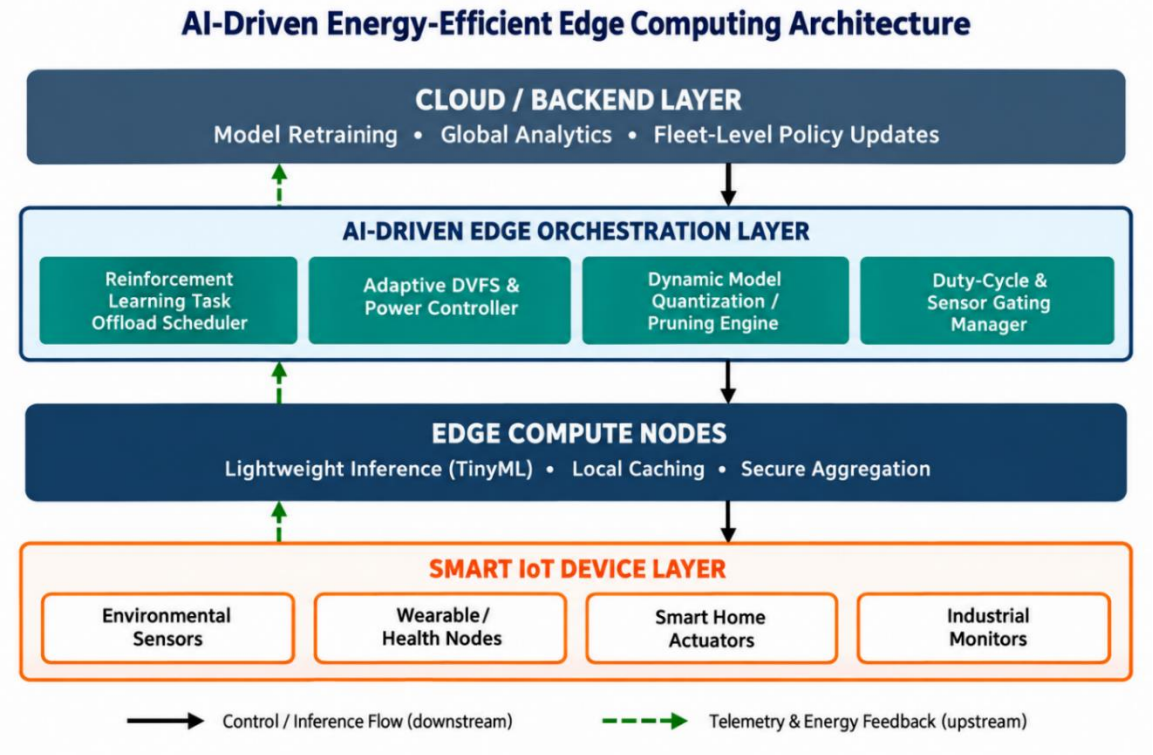


Fig. 1. Four-layer AI-driven edge computing architecture, showing downstream control/inference flow and upstream telemetry feedback.

A. Smart IoT Device Layer

This layer consists of the heterogeneous fleet of end devices, such as environmental sensors, wearable and health-monitoring nodes, smart-home actuators, and industrial monitors[26]. At the time of connection, and periodically thereafter, each device reports a lightweight capability profile (available memory, battery state, duty-cycle constraints) to its associated edge compute node.

B. Edge Compute Nodes

Edge compute nodes host lightweight inference runtimes (TinyML interpreters)[27], local result caching, and secure aggregation for any federated-learning updates. Each node runs the current model configuration and duty-cycle policy assigned by the orchestration layer, and reports execution telemetry (latency, energy draw, cache hit rate) back upstream, a pattern broadly consistent with recently proposed edge-as-a-service orchestration models[28].

C. AI-Driven Edge Orchestration Layer

This layer is the main contribution of the architecture and is composed of four interconnected modules:

- **Reinforcement Learning Task Offload Scheduler:** in the spirit of joint communication-and-offloading strategies studied for cooperative edge networks[29], this scheduler makes per-task decisions on where to perform inference based on task criticality, current battery level, network conditions, and learned energy-latency tradeoffs.
- **Adaptive DVFS and Power Controller:** makes a short-term prediction of upcoming tasks to select the voltage/frequency operating point of the processor, so that power scaling leads the workload rather than merely following it.
- **Dynamic Model Quantization/Pruning Engine:** selects, per task class, a suitable compressed model variant from a pre-computed energy-versus-inference-confidence Pareto set (see Section IV-B), chosen to meet the inference-confidence requirement within the available energy budget.

- **Duty-Cycle and Sensor Gating Manager:** modifies the sensor sampling frequency as a function of event salience, lowering the sampling frequency (and energy draw) during quiet periods and raising it when the likelihood of an anomaly increases.

D. Cloud/Backend Layer

The cloud layer periodically retrains the model on aggregated, and if needed, differentially private telemetry, calculates fleet-level analytics, and pushes updated policy parameters and compressed model variants back down to the orchestration layer. It is deliberately kept outside the real-time control loop, running at a lower cadence of hours to days, so that system responsiveness never depends on wide-area connectivity.

4. METHODOLOGY

A. Problem Formulation

We formulate joint task offloading and resource configuration as a constrained Markov Decision Process (MDP) defined by the tuple (S, A, P, R, γ), following the general framing used in recent DRL-based offloading work[30] [31]. The state s ∈ S at each decision epoch includes device battery level, current network latency and bandwidth estimate, edge-node queue depth, task criticality class, and recent sensor event rate. The action a ∈ A is a joint decision over four sub-actions, (1) execution location ∈ {device, edge, cloud}, (2) DVFS operating point, (3) model compression level, and (4) sensor sampling interval. The reward function R is a sum of an energy cost, a latency penalty and an accuracy-shortfall penalty:

$$R(s, a) = -[\, w_1 \cdot E(s,a) + w_2 \cdot L(s,a) + w_3 \cdot \max(0, A_{min} - A(a)) \,] \qquad (1)$$

In this formula:

- E(s,a) is the energy consumed by the task under configuration a.
- L(s,a) is the resulting end-to-end latency.
- A(a) is the expected accuracy at the selected compression tier, which is lower bounded by the task-specific requirement Amin.
- w1, w2, w3 are adjustable weights balancing the tradeoffs to match system goals.

To train the scheduler, we first perform offline reinforcement learning, specifically Proximal Policy Optimization (PPO), on recorded telemetry logs. Further, after deployment the model is periodically adapted with policy updates, broadcast from the cloud.

B. Dynamic Model Compression

To map the tradeoffs, we first profile an offline accuracy-energy Pareto frontier for each task class, based on the compression taxonomies in recent TinyML surveys[16], [32]. We generate candidate models by sweeping structured channel pruning ratios in combination with 8-bit and 4-bit integer quantization. The joint search over compression and scheduling parameters is performed in a similar multi-objective spirit as recent Bayesian-optimization-plus-RL methods for TinyML deployment[33]. At runtime, the orchestrator selects the most energy-efficient model that can satisfy the accuracy requirements. If battery levels start to sag, the system will swap in a leaner, more aggressively compressed variant dynamically to extend operational life, as long as performance stays above the task's acceptable floor.

C. Adaptive DVFS Control

The DVFS controller looks ahead rather than responding to CPU utilization spikes[34]. It takes a short-horizon forecast of incoming tasks directly from the scheduler and selects a clock speed that minimizes the energy-delay product without breaching per-task latency deadlines. Planning ahead this way reduces the frequency of voltage and clock hops, a meaningful optimization since every state change incurs its own energy and latency penalty.

D. Duty-Cycled Sensor Gating

The sensor sampling interval is modulated using an exponentially weighted moving average of recent event-salience scores, derived from lightweight on-device change detection, in a manner conceptually

related to recent deep-reinforcement-learning approaches to adaptive sensor coverage and scheduling in wireless sensor networks[35]. Periods of low variability trigger reduced sampling rates, subject to a domain-specific floor, while detected anomalies trigger a temporary burst of high-frequency sampling so transient events are not missed. Because battery degradation itself changes the achievable sampling budget over a device's lifetime, the gating policy is also informed by longer-horizon battery-health projections of the kind explored in recent DRL-based battery-management studies for sensor networks[36].

5. EXPERIMENTAL SETUP

A controlled, large-scale field deployment across thousands of physical devices was outside the scope of this study, so evaluation was carried out using a discrete-event simulation framework calibrated against publicly documented power profiles for representative embedded-inference hardware: Cortex-M class microcontrollers, Cortex-A class edge SoCs, and cloud GPU inference instances. Simulation parameters are summarized in Table II. The results reported in Section VI should therefore be read as simulation-based projections that illustrate the relative behavior of the proposed architecture against its baselines, rather than as measurements taken from physical field hardware.

Table 2: Simulation Configuration and Evaluation Parameters

Parameter	Value / Range
Simulated IoT device classes	Environmental sensor, wearable node, smart-home actuators, industrial monitor.
Edge node compute class	Quad-core Cortex-A, 4 GB RAM, NPU accelerator
Cloud inference class	GPU-backed inference instance
Battery capacity (device)	1200 mAh, 3.7 V nominal
Concurrent device range	10 - 200 devices per edge node
Model families evaluated	CNN classifier (image/vibration), RNN classifier (time-series)
Compression levels evaluated	1x (baseline FP32) to 16x (INT4 + structured pruning)
RL algorithm	Proximal Policy Optimization (PPO)
Training traces	Synthetic workload traces, 30-day simulated horizon
Baselines compared	Cloud-only offload; static edge (fixed configuration, no AI scheduler)

6. RESULTS AND DISCUSSIONS

This section reports the simulated performance of the proposed AI-driven edge architecture against two baselines: (1) cloud-only offload, in which all inference tasks are sent to a cloud endpoint, and (2) static edge, in which inference runs locally at the edge node using a fixed model configuration and a fixed sensor sampling rate, with no adaptive scheduling.

A. Energy Consumption

Fig. 2 shows the average energy consumed per inference for each configuration. The proposed AI-driven edge architecture averages 41.8 mJ per inference, compared with 97.2 mJ for static edge and 186.4 mJ for cloud-only offload, a 77.6% reduction relative to cloud-only offload and a 57.0% reduction relative to static edge. The largest contributors to this reduction are the drop in radio transmission energy for tasks that can be resolved locally, and the use of compressed model variants that cut compute cycles while still meeting accuracy requirements.

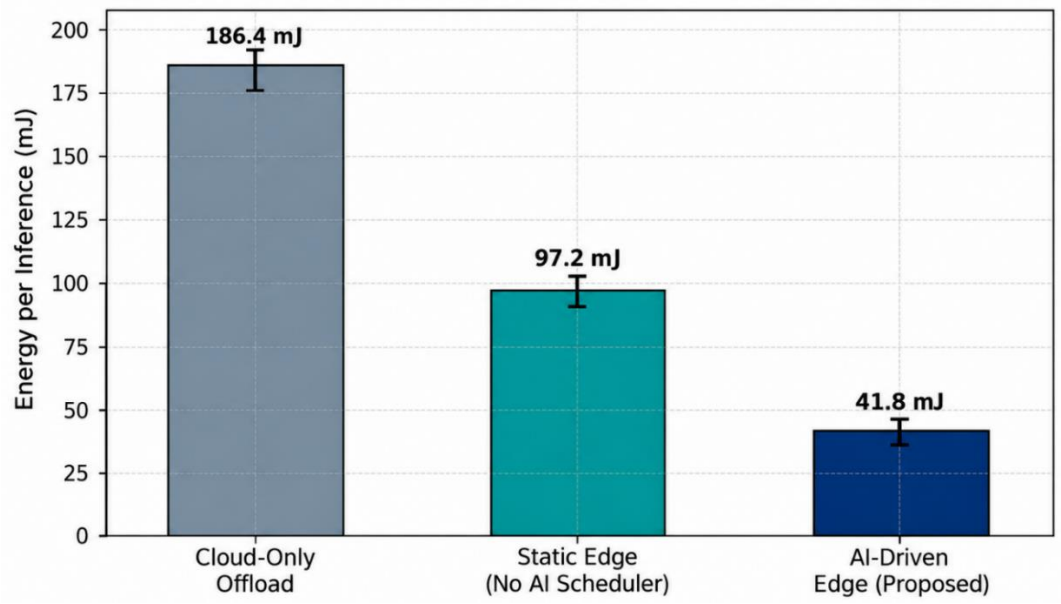


Fig. 2. Average energy consumption per inference across deployment architectures (error bars show one standard deviation across simulated device classes).

B. Latency Under Increasing Device Load

Fig. 3 shows end-to-end latency as the number of concurrent devices per edge node rises from 10 to 200. Cloud-only offload shows the steepest growth, driven by backhaul contention, climbing from roughly 48 ms to 166 ms. Static edge scales more gracefully but still grows linearly with load, since it has no way to redistribute tasks intelligently. The AI-driven edge architecture has the flattest curve among the three and stays below 35 ms at 200 concurrent devices. This is due to the RL scheduler anticipating and offloading some portion of the low-criticality tasks to the neighboring underutilized resources before the queue depth becomes a bottleneck.

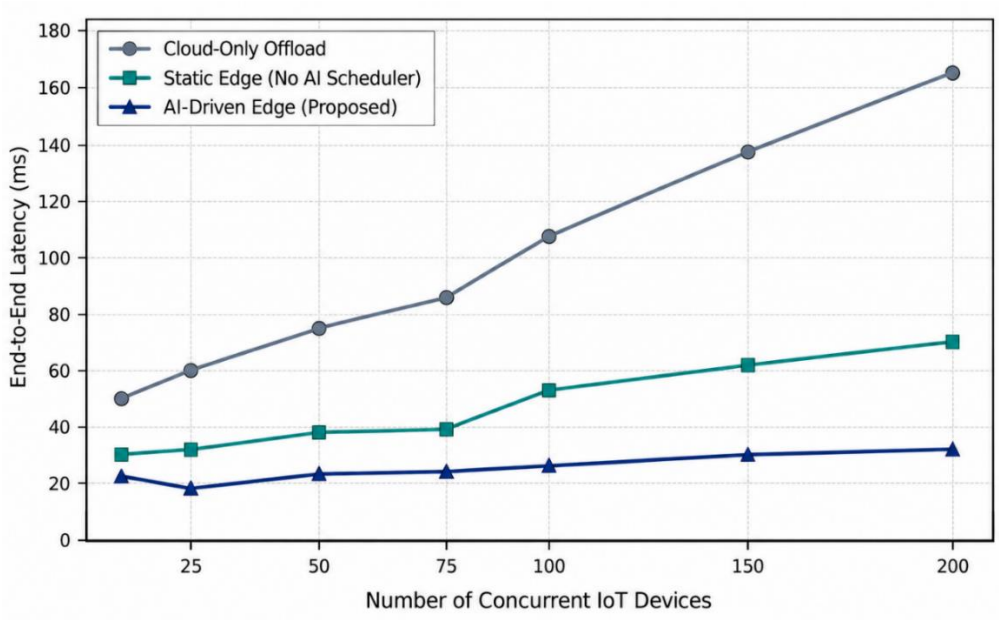


Fig. 3. Inference latency versus number of concurrent IoT devices sharing an edge node.

C. Accuracy-Energy Tradeoff

Fig. 4 shows the tradeoff between model compression ratio, resulting inference accuracy, and energy savings compared to the uncompressed (FP32) model. Accuracy degrades gracefully up to ~8x compression (93.8% accuracy, 58% energy savings). Beyond this point, further compression causes a disproportionate drop in accuracy. This defines the 4x-8x operating band shown in the figure, which the dynamic compression engine treats as the default search space for jobs with moderate accuracy demands, resorting to more aggressive compression only for low-criticality task classes.

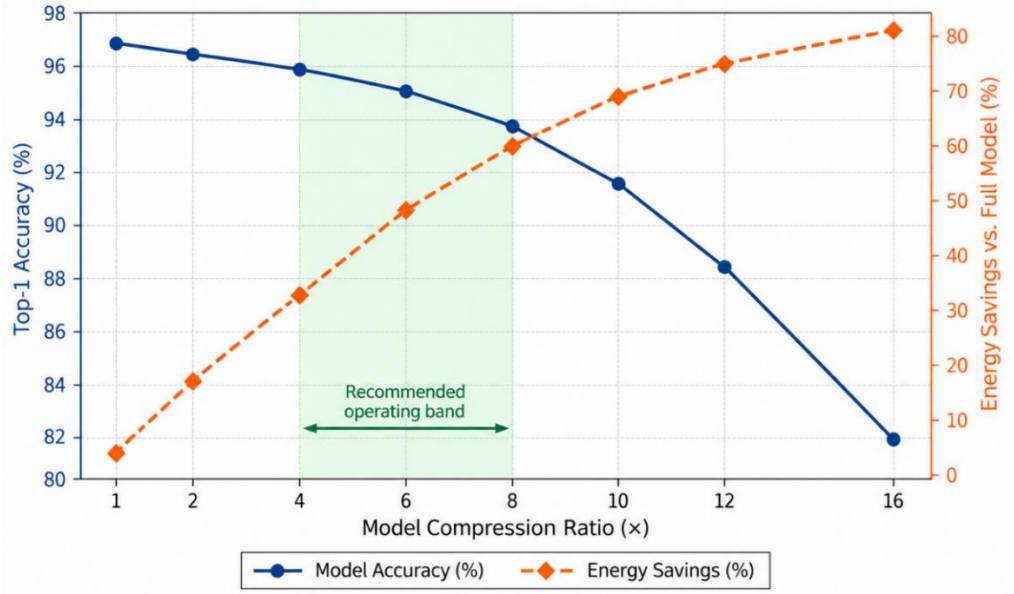


Fig. 4. Accuracy-energy tradeoff under adaptive model compression, with the recommended default operating band highlighted.

D. Battery Life Impact

Fig. 5 translates per-inference energy savings into projected device battery life under a representative continuous-monitoring workload for a 1200 mAh cell. Projected runtime rises from 14.2 hours under continuous cloud offload, to 26.5 hours under static edge, to 52.7 hours under the proposed AI-driven edge architecture, a 3.7x improvement over cloud-only offload and a 2.0x improvement over static edge. This is consistent with the combined effect of reduced radio usage, compressed inference, and duty-cycled sensing working together.

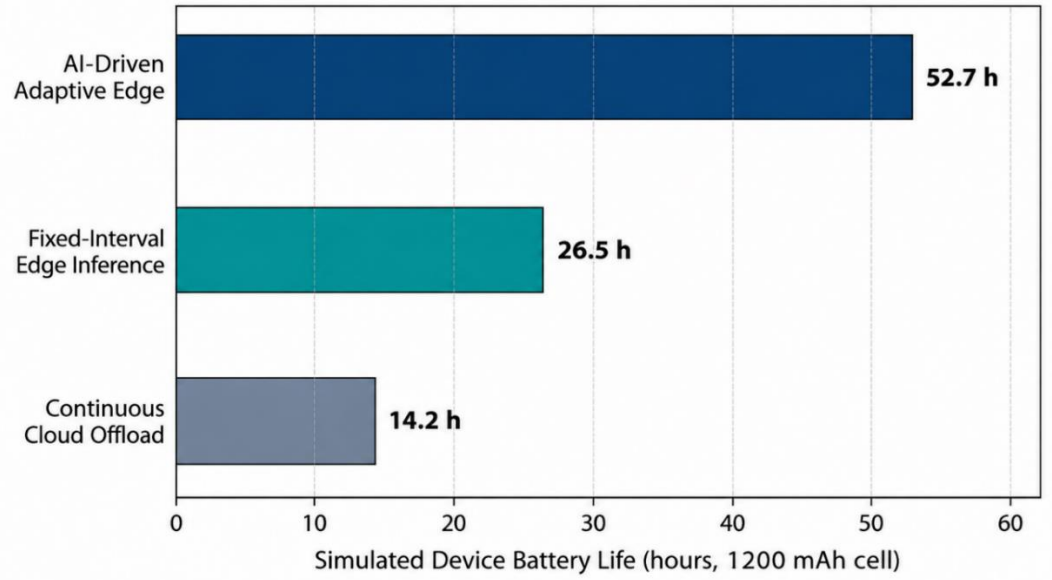


Fig. 5. Projected device battery life under each architecture, assuming a representative continuous-monitoring workload.

E. Ablation Study

To isolate the contribution of each AI-driven module, we enabled components incrementally, starting from the static-edge baseline of 97.2 mJ/inference. Fig. 6 shows that the RL task-offload scheduler alone accounts for the largest single reduction (19.2%, down to 78.4 mJ). Adding dynamic model quantization on top of offloading contributes the next-largest share, bringing the figure down to 61.4 mJ, followed by a smaller reduction from adaptive DVFS control, down to 57.8 mJ. Finally, duty-cycled sensor gating brings the total down to the final 41.8 mJ figure, primarily by suppressing redundant inference calls during low-activity periods rather than by lowering the cost of any single inference.

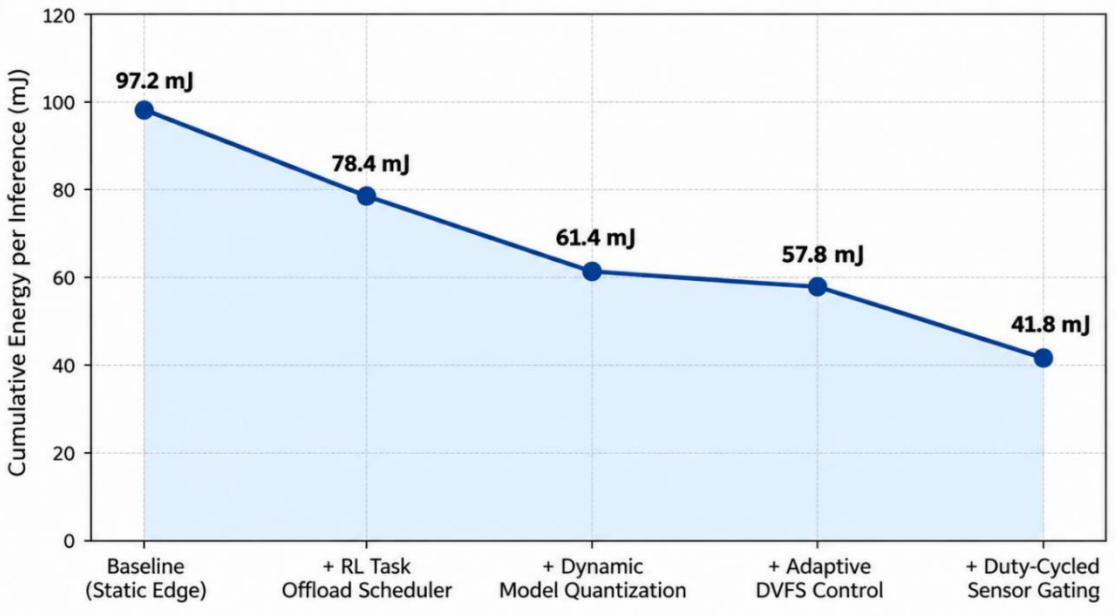


Fig. 6. Ablation study of the cumulative energy reduction when enabling each AI-driven module progressively, starting from the static-edge baseline.

F. Summary Comparison

Table 3 summarizes the main metrics discussed above. AI-driven edge accuracy is a workload-weighted average of dynamically selected compression levels, not a single static model.

Table 3: Summary Comparison of Key Performance Metrics Across the Three Evaluated Architectures

Metric	Cloud-Only	Static Edge	AI-Driven Edge (Proposed)	Improvement vs. Static Edge
Energy per inference (mJ)	186.4	97.2	41.8	-57.0%
Median latency (ms, 100 devices)	112	46	27	-41.3%
Projected battery life (h)	14.2	26.5	52.7	+98.9%
Model accuracy (task-avg., %)	96.8	95.9	95.1*	-0.8 pp
Bandwidth consumption (rel.)	1.00x	0.31x	0.14x	-54.8%

7. SCALABILITY, SECURITY, AND DEPLOYMENT CONSIDERATIONS

A. Scalability

The architecture is built for horizontal scaling. Instead of overloading a single centralized controller, we simply add edge nodes, each with a bounded cluster of devices to manage, consistent with the horizontally scalable edge-as-a-service patterns described in recent orchestration literature[28]. Each local orchestrator executes the RL policy as a fast, lightweight forward pass, sending only aggregated telemetry, never raw sensor streams, upstream for periodic retraining. This keeps backhaul bandwidth and compute requirements growing linearly, rather than quadratically, with overall fleet size.

B. Security and Privacy

Processing inference at the edge keeps raw sensor feeds off the network. This significantly reduces the attack surface for eavesdropping and protects sensitive information such as occupancy trends or personal biometrics. Where edge nodes do communicate back to the cloud, the architecture employs secure aggregation to protect federated model weights, based on hierarchical, privacy-preserving fog federated learning designs proposed for IoT intrusion detection [37]. For compliance requiring stronger

protections, differential-privacy noise can be injected directly into the telemetry logs used for retraining scheduling policies. Reinforcement-learning-based adaptive encryption schemes recently proposed for wireless sensor networks provide a complementary layer of defense against link-level cyberthreats [38]. Personalized, energy-aware model-decoupling techniques for federated cloud-edge networks also indicate that privacy and per-device energy efficiency can be optimized together, rather than traded off [39]. Combining federated learning with explainable-AI methods can further help operators audit what a locally retrained policy has learned before it is redeployed [40]. Analogous to secure federated fine-tuning schemes proposed for heterogeneous edge deployments, edge nodes and devices should authenticate using mutually verified certificates, and any over-the-air policy or model update from the cloud layer should be signed to prevent tampering[41].

C. Deployment Considerations

The reward-function weights (w1, w2, w3) should be calibrated for practical deployment to application-specific priorities, e.g., weighting latency more heavily for safety-critical industrial monitoring than for periodic environmental logging. Policies should be initially trained on representative historical or synthetic traces before field deployment. During the initial period of online fine-tuning conservative fallback rules should be used, e.g., falling back to a fixed and known safe configuration, so that unstable early-stage RL behavior does not impact production workloads. Where retraining occurs across many devices, green federated-learning strategies aimed at reducing the carbon and energy cost of the aggregation process itself are a useful complement to the on-device savings described in Section 6 [42].

8. CONCLUSION AND FUTURE WORK

This paper presented an AI-driven, energy-efficient edge computing architecture for smart IoT devices, coordinating RL-based task offloading, dynamic DVFS control, dynamic model quantization, and duty-cycled sensor gating within a single four-layer system. Using representative embedded power profiles, our simulations show substantial energy savings, 77.6% relative to cloud-only offload and 57.0% relative to static edge, alongside a modest 0.8 percentage-point drop in workload-average model accuracy, a 3.7x projected extension in device battery life, and latency that stays low and scales gracefully as device load increases.

Future work will proceed along four lines: (i) validating these simulation-based projections on physical hardware testbeds across a range of device classes, in line with recent calls for hardware-level evaluation of on-device AI models; (ii) extending the orchestration layer to support federated on-device learning, so that models can adapt locally at edge nodes without transmitting raw telemetry, building on recent personalized and energy-aware federated-learning designs for cloud-edge networks ; (iii) studying how well the learned scheduling policy generalizes across different, heterogeneous IoT deployments; and (iv) contributing to standardized, cross-layer, energy-aware orchestration interfaces, informed by green federated-learning practices [21], so that the proposed architecture can interoperate with third-party edge platforms, consistent with the broader push toward distributed, model-optimized edge-cloud collaboration.

DECLARATIONS:

Acknowledgments	:	Not applicable.
Conflict of Interest	:	The author declares that there is no actual or potential conflict of interest about this article.
Consent to Publish	:	The authors agree to publish the paper in the Global Research Journal of Social Sciences and Management.
Ethical Approval	:	Not applicable.
Funding	:	Author claims no funding was received.
Author Contribution	:	Both the authors confirms their responsibility for the study, conception, design, data collection, and manuscript preparation.
Data Availability Statement	:	The data presented in this study are available upon request from the corresponding author.

Use of Generative Artificial Intelligence	:	AI-assisted tools have been used to improve language editing or grammatical clarity, the authors take full responsibility for the content of the final version.

REFERENCES

[1] V. Upadrista, "The IoT Standards Reference Model," Apress, 2021, pp. 61–86. doi: 10.1007/978-1-4842-7271-8_4.

[2] A. Pandey and P. Chandani, "PRACTICAL AND INNOVATIVE APPLICATIONS OF IOT AND IOT NETWORKS," Iterative International Publishers Selfypage Developers Pvt Ltd, 2024, pp. 146–155. doi: 10.58532/v3bdio3p2ch2.

[3] X. Zhang and Q. Zhu, "Statistical Tail-Latency Bounded QoS Provisioning for Parallel and Distributed Data Centers," Institute Of Electrical Electronics Engineers, July 2021, pp. 762–772. doi: 10.1109/icdcs51616.2021.00078.

[4] S. Annam, "Strategies for Data Privacy in Telecommunication Systems," IJSRCSEIT, pp. 397–406, Nov. 2023, doi: 10.32628/cseit2390681.

[5] C. Arendt, S. Böcker, J. Tiemann, M. Patchou, and C. Wietfeld, "Pushing the Limits: Resilience Testing for Mission-Critical Machine-Type Communication," July 21, 2021, European Society for Socially Embedded Technologies. doi: 10.48550/arxiv.2107.10065.

[6] T. Widmann, F. Merkle, M. Nocker, and P. Schöttle, "Pruning for Power: Optimizing Energy Efficiency in IoT with Neural Network Pruning," Springer Nature Switzerland, 2023, pp. 251–263. doi: 10.1007/978-3-031-34204-2_22.

[7] E. Oikonomou, A. Rouskas, and D. N. Skoutas, "Adaptive Node Management in Edge Networks Under Time-Varying Workloads," Institute Of Electrical Electronics Engineers, Dec. 2024, pp. 1–6. doi: 10.1109/vcc63113.2024.10914479.

[8] [8] S. Zhu, K. Ota, and M. Dong, "Energy-Efficient Artificial Intelligence of Things With Intelligent Edge," IEEE Internet Things J., vol. 9, no. 10, pp. 7525–7532, May 2022, doi: 10.1109/jiot.2022.3143722.

[9] J.-C. Charr, R. Couturier, A. Fanfakh, and A. Giersch, "Energy Consumption Reduction with DVFS for Message Passing Iterative Applications on Heterogeneous Architectures," Institute Of Electrical Electronics Engineers, May 2015, pp. 922–931. doi: 10.1109/ipdpsw.2015.44.

[10] S. Zhuo et al., "An Empirical Study of Low Precision Quantization for TinyML," Mar. 10, 2022, European Society for Socially Embedded Technologies. doi: 10.48550/arxiv.2203.05492.

[11] I. Jemili, D. Ghrab, A. Belghith, and M. Mosbah, "Context-Aware Broadcast in Duty-Cycled Wireless Sensor Networks," International Journal on Semantic Web and Information Systems, vol. 13, no. 3, pp. 48–67, July 2017, doi: 10.4018/ijswis.2017070103.

[12] A. Agarwal, "Edge Computing and Fog Architecture: Reshaping Distributed System Boundaries," World J. Adv. Eng. Technol. Sci., vol. 15, no. 3, pp. 827–838, June 2025, doi: 10.30574/wjaets.2025.15.3.1004.

[13] Q. Li, M. Guo, Z. Peng, D. Cui, and J. He, "Edge–Cloud Collaborative Computation Offloading for Mixed Traffic," IEEE Systems Journal, vol. 17, no. 3, pp. 5023–5034, Sept. 2023, doi: 10.1109/jsyst.2023.3277003.

[14] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, and S. Han, "Tiny Machine Learning: Progress and Futures," Mar. 28, 2024, European Society for Socially Embedded Technologies. doi: 10.48550/arxiv.2403.19076.

[15] R. Immonen and T. Hämäläinen, "Tiny Machine Learning for Resource-Constrained Microcontrollers," Journal of Sensors, vol. 2022, pp. 1–11, Nov. 2022, doi: 10.1155/2022/7437023.

[16] S. Heydari and Q. H. Mahmoud, "Tiny Machine Learning and On-Device Inference: A Survey of Applications, Challenges, and Future Directions.," Sensors, vol. 25, no. 10, p. 3191, May 2025, doi: 10.3390/s25103191.

[17] R. Agrawal, H. Kumar, and S. R. Lnu, "Efficient LLMs for Edge Devices: Pruning, Quantization, and Distillation Techniques," Institute Of Electrical Electronics Engineers, Mar. 2025, pp. 1413–1418. doi: 10.1109/icmlas64557.2025.10968787.

[18] D. T. P. Chander, "Optimizing Memory Efficiency in Large Language Models: Adaptive Compression Techniques," IJRASET, vol. 13, no. 1, pp. 1474–1479, Jan. 2025, doi: 10.22214/ijraset.2025.66591.
[19] N. Din, H. Chen, and D. Khan, "Mobility-Aware Resource Allocation in Multi-Access Edge Computing Using Deep Reinforcement Learning," Institute Of Electrical Electronics Engineers, Dec. 2019, pp. 202–209. doi: 10.1109/ispa-bdcloud-sustaincom-socialcom48970.2019.00038.
[20] A. Uddin, A. Sakr, and N. Zhang, "Intelligent Offloading in Vehicular Edge Computing: A Comprehensive Review of Deep Reinforcement Learning Approaches and Architectures," Feb. 10, 2025, American Chemical Society. doi: 10.48550/arxiv.2502.06963.
[21] S. A. Ullah et al., "From Nodes to Roads: Surveying DRL Applications in MEC-Enhanced Terrestrial Wireless Networks," IEEE Commun. Surv. Tutorials, vol. 28, pp. 1169–1208, Jan. 2026, doi: 10.1109/comst.2025.3568035.
[22] C. Wang, Y. Guan, S. Peng, H. Chen, and G. Li, "Data prioritization aware resource allocation in internet of vehicles using multi-agent deep reinforcement learning.," Neural Networks, vol. 190, p. 107671, Oct. 2025, doi: 10.1016/j.neunet.2025.107671.
[23] F. Yıldırım, Y. Yalman, K. Ç. Bayındır, and E. Terciyanlı, "Comprehensive Review of Edge Computing for Power Systems: State of the Art, Architecture, and Applications," Applied Sciences, vol. 15, no. 8, p. 4592, Apr. 2025, doi: 10.3390/app15084592.
[24] J. Liu et al., "Edge-Cloud Collaborative Computing on Distributed Intelligence and Model Optimization: A Survey," IEEE Commun. Surv. Tutorials, vol. 28, p. 1, Jan. 2026, doi: 10.1109/comst.2026.3669216.
[25] V. S, V. G, and A. F, "Artificial Intelligence in Edge Computing and IoT Devices: A Comprehensive Survey on Distributed Intelligence," IEJCSA, vol. 4, no. 1, pp. 8–23, Feb. 2026, doi: 10.34293/iejcsa.v4i1.68.
[26] A. Sharma and N. Singh, "Sensors, Embedded Systems, and IoT Components," Chapman Hall Crc, 2022, pp. 1–15. doi: 10.1201/9781003291916-1.
[27] S. Salerno, "Tiny Machine Learning," in Maker Innovations Series, Apress, 2025, pp. 1–40. doi: 10.1007/979-8-8688-1294-1_1.
[28] A. Jalal et al., "Empowering Edge Computing: Public Edge as a Service for Performance and Cost Optimization," CMC, vol. 86, no. 2, pp. 1–19, Jan. 2026, doi: 10.32604/cmc.2025.068289.
[29] Y. Li, Z. Yi, D. Guo, L. Luo, B. Ren, and Q. Zhang, "Joint Communication and Offloading Strategy of CoMP UAV-Assisted MEC Networks," IEEE Internet Things J., vol. 12, no. 19, pp. 39788–39802, Oct. 2025, doi: 10.1109/jiot.2025.3588840.
[30] L.-H. Zhao et al., "Energy-Efficient Task Offloading in Multi-server Mobile Edge Computing Networks: A Deep Reinforcement Learning Approach," Arab J Sci Eng, vol. 50, no. 19, pp. 16221–16242, June 2025, doi: 10.1007/s13369-025-10363-y.
[31] P. Kansal, M. Kumar, and O. P. Verma, "Edge-Cloud based Framework to Offload Task in Vehicular Edge Computing Using Optimized Deep Reinforcement Learning Approaches," Institute Of Electrical Electronics Engineers, Dec. 2024, pp. 1–7. doi: 10.1109/ccis63231.2024.10932058.
[32] S. Somvanshi et al., "From Tiny Machine Learning to Tiny Deep Learning: A Survey," June 21, 2025. doi: 10.48550/arxiv.2506.18927.
[33] M. Deutel, G. Kontes, C. Mutschler, and J. Teich, "Combining Multi-Objective Bayesian Optimization with Reinforcement Learning for TinyML," ACM Trans. Evol. Learn. Optim., vol. 5, no. 3, pp. 1–21, Aug. 2025, doi: 10.1145/3715012.
[34] H. Noguchi and M. Kaneko, "Energy-Efficient CPU Core Allocation in DVFS and CPU Pinning Enabled Cloud," Institute Of Electrical Electronics Engineers, June 2024, pp. 67–72. doi: 10.1109/cloud-summit61220.2024.00018.
[35] P. Zhou, M. Kan, W. Chen, Y. Wang, and B. Cao, "An adaptive coverage method for dynamic wireless sensor network deployment using deep reinforcement learning," Sci Rep, vol. 15, no. 5, p. 30304, Aug. 2025, doi: 10.1038/s41598-025-16031-3.
[36] J.-H. Jeong, H. Jo, Q. Zhou, T. Nishat, and L. Wu, "Active management of battery degradation in wireless sensor network using deep reinforcement learning for group battery replacement,"

Mar. 20, 2025, European Society for Socially Embedded Technologies. doi: 10.48550/arxiv.2503.15865.
[37] M. M. Islam, W. M. Abdullah, and B. N. Saha, "Privacy-Preserving Hierarchical Fog Federated Learning (PP-HFFL) for IoT Intrusion Detection," Oct. 14, 2025, Mdpi Ag. doi: 10.20944/preprints202510.1115.v1.
[38] S. B. N. Premakumari, G. Sundaram, M. Rivera, P. Wheeler, and R. E. P. Guzmán, "Reinforcement Q-Learning-Based Adaptive Encryption Model for Cyberthreat Mitigation in Wireless Sensor Networks," Sensors, vol. 25, no. 7, p. 2056, Mar. 2025, doi: 10.3390/s25072056.
[39] C. Jin, T. Du, and X. Chen, "Energy-Efficient Model Decoupling for Personalized Federated Learning on Cloud-Edge Computing Networks," Trans Emerging Tel Tech, vol. 36, no. 7, June 2025, doi: 10.1002/ett.70203.
[40] P. Dubey and M. Kumar, "Integrating Explainable AI with Federated Learning for Next-Generation IoT: A comprehensive review and prospective insights," Computer Science Review, vol. 56, p. 100697, May 2025, doi: 10.1016/j.cosrev.2024.100697.
[41] Z. Gao, Z. Zhang, Y. Guo, and Y. Gong, "Federated Adaptive Fine-Tuning of Large Language Models with Heterogeneous Quantization and LoRA," Institute Of Electrical Electronics Engineers, May 2025, pp. 1–10. doi: 10.1109/infocom55648.2025.11044641.
[42] D. Thakur, A. Guzzo, G. Fortino, and F. Piccialli, "Green Federated Learning: A New Era of Green Aware AI," ACM Comput. Surv., vol. 57, no. 8, pp. 1–36, Mar. 2025, doi: 10.1145/3718363.

Authors' Profiles



Sayamuddin Ahmed Jilani completed his BTech in CSE from MAKAUT, WB(formerly known as WBUT) in 2010 and MTech in Multimedia and Software Systems under CSE department from NITTTR-Kolkata in 2013, registered in PhD under MAKAUT, WB; he started his career in 2013 as a Registered Councelor in PIMT (An IGNOU Study Centre) for teaching BCA and MCA Students. He moved to professional teaching in 2014, he joined St. Mary's Technical Campus as an Assistant Professor in the Department of CSE. He appeared in various FDP. He has been awarded the Maulana Azad National Fellowship for Pursuing PhD. He has over 10 years of Teaching Experience. His research interests includes Artificial Intelligence, Internet of Things, Wireless Sensor Networks.



Dr. Soumitra Kumar Mandal has obtained his B.E. from Bengal Engineering College (Now IIEST), Shibpur, M.Tech from Institute of Technology, Banaras Hindu University, Varanasi and Ph.D. from Punjab University, Chandigarh all in Electrical Engineering. He started his career as Lecturer at SSGM Engineering College, Shegaon, and then moved to Punjab Engineering College, Chandigarh. In February, 2004, he has been appointed as Assistant Professor of Electrical Engineering in National Institute of Technical Teachers' Training and Research (NITTTR), Kolkata. He is now serving as Professor of Electrical Engineering in the same institute. Throughout his academic career, he has published about 45 research papers in National and International Journals and presented many papers in conferences. He has also published 8 Text books for undergraduate and Post Graduate Students of Electrical Engineering. His research interests include Microprocessor and Microcontroller based System Design, Embedded System Design, Computer Controlled Drives, Neuro-fuzzy Computing, Signal Processing and VLSI design. He is also a life member of ISTE and a member of IE.