Computers and Artificial Intelligence；Vol.3, No.4; 2026
E-ISSN: 3006 -3728 P-ISSN：3006-371X
DOI: https://doi.org/10.70267/cai.26v3n4.1628
Published by: Zeus Press

Machine Learning-Enabled Edge Intelligence for IoT Communication Systems: A Structured Review

Chen Huang*

School of Information, Xiamen University, Xiamen Fujian, 361102, P.R.China

**Corresponding author: Chen Huang.*

Abstract

This structured review investigates how machine learning-enabled edge intelligence can improve the performance, efficiency, and resilience of Internet of Things communication systems under constraints of latency, bandwidth, energy, privacy, and device heterogeneity. A structured literature review was conducted using peer-reviewed studies from major academic databases, with the selected work classified according to learning paradigms, edge deployment strategies, communication functions, application domains, evaluation metrics, and practical limitations. The synthesis shows that supervised learning, deep learning, reinforcement learning, federated learning, lightweight model compression, and Tiny Machine Learning can support adaptive scheduling, intelligent routing, computation offloading, bandwidth allocation, anomaly detection, and privacy-preserving collaboration across device-edge-cloud architectures. The reviewed evidence indicates that these approaches can reduce end-to-end delay, communication traffic, and energy consumption while improving resource utilization, local autonomy, and responsiveness in smart cities, industrial systems, healthcare, agriculture, and intelligent transportation. However, performance gains remain strongly dependent on communication conditions, model size, data distribution, hardware capability, synchronization overhead, and security requirements. The review further identifies unresolved challenges involving non-independent and identically distributed data, unstable wireless links, model adaptability, privacy leakage, adversarial threats, and the absence of standardized multi-objective benchmarks. It concludes that future Internet of Things systems should adopt communication-computation-learning co-design, lightweight and adaptive models, privacy-aware distributed intelligence, and cross-layer orchestration to achieve scalable, trustworthy, and energy-efficient edge intelligence.

Keywords

edge intelligence, internet of things communication, machine learning, federated learning, computation offloading

1. Introduction

1.1 Background and Motivation

Contemporary IoT deployments connect cameras, industrial controllers, wearables, vehicles, and low-power sensors through a mix of embedded and wireless technologies. Their data streams support traffic management, factory automation, remote health monitoring, and intelligent transportation [1, 2].

In cloud-centered architectures, much of this data is forwarded to remote servers. The round trip adds delay and backhaul traffic, while workloads such as video analytics and anomaly detection may exceed both device capacity and the available network budget [3, 4]. Centralizing raw measurements also transfers sensitive personal and operational data outside the local domain. These weaknesses are most visible when a control action or warning must be produced within a fixed deadline.

Edge intelligence places selected inference, training, and optimization functions at gateways, base stations, roadside units, industrial controllers, or small edge servers. A camera may send features rather than full frames; a factory gateway may give fault packets priority; a roadside server may coordinate offloading with radio scheduling. Each choice reduces cloud traffic but consumes shared radio, memory, and compute resources at the edge [6, 8, 11, 45].

This review treats ML-enabled EI as a joint communication, computation, and learning problem. Scheduling, routing, and offloading depend on channel quality, queue state, deadlines, battery level, and model size. An accurate model may still be a poor deployment choice when it produces excessive uplink traffic or synchronization delay. The analysis therefore follows the communication functions changed by learning under practical edge constraints.

1.2 Research Gap and Problem Statement

Several surveys describe the convergence of edge computing and deep learning, general EI taxonomies, and application-oriented AIoT systems [6, 7, 9, 13]. Their organizing frameworks usually emphasize architectures, enabling technologies, or use cases. Direct comparisons of learning methods for scheduling, routing, bandwidth allocation, and computation offloading receive less sustained treatment [12, 13].

The unresolved issue is the link between a learning choice and its communication cost. Device heterogeneity, model size, local data distributions, privacy controls, and coordination overhead can change the outcome of the same algorithm across deployments [20, 22]. This review focuses on that link, comparing technical mechanisms together with the conditions that limit them.

1.3 Objectives and Contributions

The review pursues three related aims:

- Relate supervised, unsupervised, deep, reinforcement, federated, and TinyML methods to specific IoT communication functions.
- Compare deployment choices across devices, gateways, MEC servers, and clouds, with attention to latency, bandwidth, energy, privacy, and model-update cost.
- Synthesize evidence from application studies and identify the benchmarking, security, and adaptability problems that remain open.

Section 2 describes the review method. Sections 3 and 4 connect edge placement with learning and communication mechanisms. Section 5 compares application settings, and Section 6 discusses unresolved technical questions before the conclusion.

Table 1: Comparison with existing surveys

Survey	Primary scope	IoT communication focus	ML/edge deployment details	Relation to the present review
Wang et al. [6]	Edge computing and deep learning convergence	Moderate; focuses on edge/deep-learning systems broadly	Discusses edge DL frameworks and deployment issues	Focuses the comparison on scheduling, routing, and offloading in IoT networks.

Survey	Primary scope	IoT communication focus	ML/edge deployment details	Relation to the present review
Xu et al. [7]	General edge intelligence taxonomy	Limited explicit treatment of IoT communication protocols and KPIs	Covers EI concepts across device-edge-cloud tiers	Uses the taxonomy as a baseline, then adds communication constraints and KPIs.
Baccour et al. [9]	Pervasive AI for IoT applications	Application-oriented rather than mechanism-oriented	Covers AIoT use cases and system requirements	Examines trade-offs among latency, bandwidth, privacy, and update synchronization.
Chang et al. [13]	Edge-computing-powered AIoT	Discusses AIoT but less focused on routing/scheduling/offloading details	Highlights AI-enabled edge computing architectures	Maps ML paradigms to scheduling, routing, offloading, and other communication functions.
Amin & Hossain [16]	Healthcare IoT and edge intelligence	Domain-specific; focuses on healthcare scenarios	Covers privacy-sensitive medical edge analytics	Contrasts healthcare requirements with those in other IoT domains.
Gong et al. [17]	Edge intelligence for transportation systems	Strong in vehicular and ITS communication scenarios	Focuses on transportation-specific edge intelligence	Extends transportation-specific findings across four application settings.
This review	ML-enabled EI for IoT communication systems	Central focus	Links FL, RL, TinyML, compression, privacy, and offloading to communication KPIs	Mechanism-level synthesis with two comparison tables and a device-edge-cloud architecture.

2. Review Method

We searched IEEE Xplore, ScienceDirect, the ACM Digital Library, MDPI, and SpringerLink for peer-reviewed work published mainly between 2020 and 2026. Queries paired "edge intelligence," "Internet of Things," and "machine learning" with terms tied to communication and deployment, including federated learning, TinyML, task offloading, adaptive scheduling, intelligent routing, distributed learning, and lightweight inference.

Screening retained journal articles and conference papers that proposed, analyzed, or experimentally evaluated an EI method for IoT communication or edge resource management. Editorials, short abstracts, studies outside the topic, and papers without a clear method or evaluation were excluded. The retained studies were coded by learning paradigm, edge tier, communication function, application domain, evaluation metric, and reported limitation; this coding guides the comparisons in Tables 1 and 2. Because the search was limited to English-language publications in selected databases, the synthesis may omit recent or less visible work, and its conclusions remain tied to the experimental settings reported by the original studies.

3. Foundations of Edge Intelligence and IoT

3.1 IoT Architecture and Edge Placement

The familiar sensing-network-application model still provides a useful map of IoT data flow [1, 2]. Sensors and actuators observe the physical environment, communication links carry measurements and control messages, and application software turns processed data into domain-specific actions. Edge computing inserts processing at several points in this path, so the boundary between the three layers is no longer fixed.

At the sensing side, cameras, RFID tags, embedded controllers, and low-power sensors acquire raw signals [3]. Battery and memory limits often favor local preprocessing. Compressed sensing lowers the sampling

burden for sparse signals, event-triggered reporting suppresses routine transmissions, and TinyML can identify simple anomalies on a microcontroller [29, 42]. The trade-offs differ: reconstruction assumptions constrain compressed sensing, event triggers may miss weak early signals, and on-device inference consumes flash memory and energy.

Wi-Fi, cellular systems, LoRaWAN, NB-IoT, and wireless sensor networks carry these observations toward gateways and services [4]. Dense access creates contention, unstable links alter delivery time, and heterogeneous devices compete for radio and compute resources. Learning-based routing, adaptive bandwidth allocation, and edge-assisted scheduling use channel, traffic, and queue information to adjust transmission decisions [11, 12, 39]. Their results are commonly reported through delay, packet delivery, spectrum use, and energy consumption.

Application requirements determine where this processing is useful. Healthcare systems prioritize privacy and response time, factories emphasize fault detection and control continuity, and transportation systems must react to rapidly changing traffic and channel conditions [16, 17, 18]. A single edge node may therefore filter sensor data, schedule network access, and execute an application model. The three-layer architecture remains a reference model, but actual EI deployments routinely cross its boundaries [1, 3].

3.2 Edge Intelligence under Communication Constraints

Edge intelligence moves selected inference and optimization tasks closer to the source of IoT data [8, 9]. Local filtering can shorten the sensing-to-response path and reduce the volume sent to the cloud, but it shifts computation and energy use to gateways or devices. Offloading policies, cached models, and compressed networks are therefore evaluated together through end-to-end delay, accuracy, bandwidth use, and energy cost [10, 11, 12, 28].

Distributed training changes the type of information exchanged across the network. Federated, split, and decentralized learning keep raw records at local sites while sending model updates or intermediate features [13, 14]. This arrangement is useful for healthcare and industrial data, yet communication rounds, client availability, and aggregation rules can dominate the final cost [15, 25, 26].

Streaming decisions add a different constraint. An edge scheduler may choose bandwidth, transmission time, and an offloading destination from the current queue, channel, battery, and deadline state. Reinforcement learning and online optimization can adapt these choices, but a policy trained on one traffic pattern may fail under congestion or mobility. Safe exploration, compact state representations, and fallback heuristics are often needed in practical systems [45].

These workloads share the same limited resources. Local inference saves a network round trip but draws device power; distributed learning protects data locality but can saturate the uplink; rapid control is useful only while the model remains calibrated. Consequently, many designs combine model compression, communication-aware scheduling, federated optimization, and edge-cloud coordination rather than tuning one component in isolation [9, 15].

3.3 Communication Functions in EI-Enabled IoT

In an EI-enabled network, learning is embedded in decisions about when to transmit, where to forward data, and where to execute a task. The relevant state includes traffic load, link quality, queue length, device capability, and application deadlines [19, 20]. Scheduling, routing, and offloading are separate control actions, but a change in one alters the others.

Adaptive schedulers use local observations to assign radio and compute resources. Reinforcement-learning policies can map channel and queue states to transmission time, bandwidth, edge-server selection, or offloading actions [21, 22, 23]. Network-slicing controllers apply similar methods when industrial services have different latency and reliability targets [39]. The main risks are slow convergence and unstable behavior when the traffic distribution changes.

Routing policies work with topology, link quality, congestion, and node energy. Compared with a fixed shortest path, an edge-resident predictor can update a route after a link failure or sudden traffic shift [24].

Studies in sensor and vehicular networks typically compare end-to-end delay, packet delivery ratio, routing overhead, and energy use [25, 26].

Offloading determines whether computation stays on the device, moves to a nearby server, or continues to the cloud. Cameras, wearables, vehicles, and industrial sensors generate tasks with different sizes and deadlines, so the decision must include channel conditions and server load as well as model complexity [12, 45]. Gateways may reserve bandwidth for urgent tasks, compress inputs before transmission, or split a model between local and edge processors.

The three functions are coupled in operation. A scheduler controls resource access, a route determines the path and its reliability, and an offloading policy selects the processing site. Evaluations that isolate one function can miss queueing or radio costs introduced elsewhere in the pipeline. The mechanism-level comparison in this review therefore tracks communication, computation, and learning outcomes together [20, 21, 27, 28].

Figure 1: Architecture of ML-enabled edge intelligence for IoT communication systems

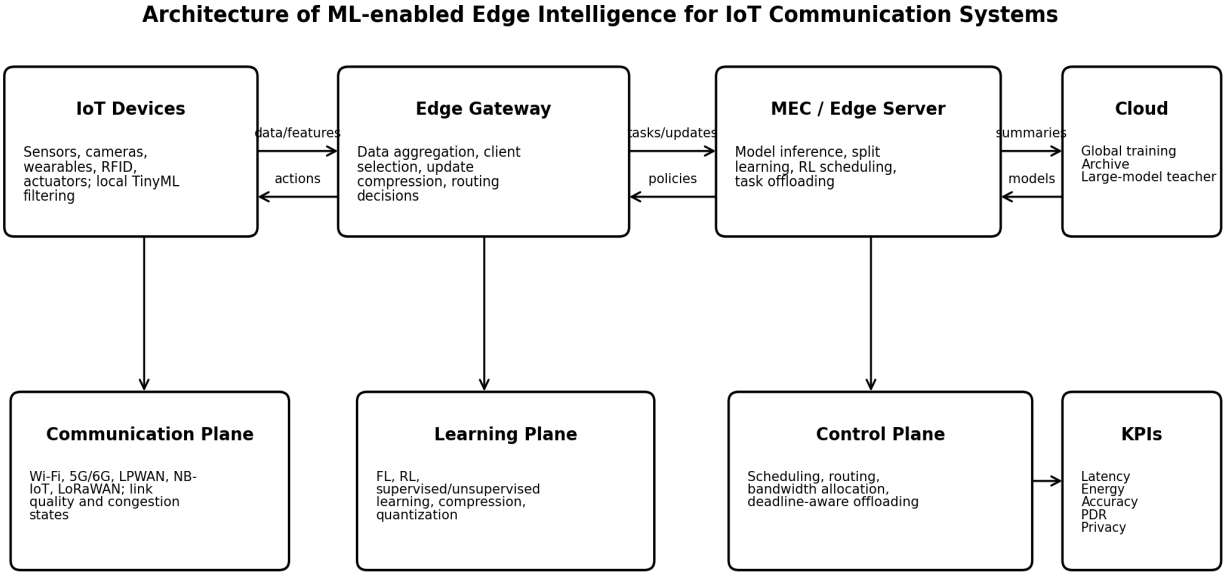

Figure 1 organizes the system into device, gateway, MEC, and cloud tiers. Devices produce observations; gateways aggregate or filter them; MEC servers handle heavier inference, coordination, and offloading; the cloud supports global training and long-term storage. Link quality, congestion, update frequency, and radio cost connect the communication, learning, and control planes to the reported latency, energy, accuracy, and privacy outcomes.

4. Machine Learning in Edge Intelligence for IoT

Machine learning at the edge is constrained by the hardware and network that carry it. Model choice affects inference time, memory use, energy draw, and the amount of information exchanged between devices and servers. The literature therefore pairs learning paradigms with compression, client selection, scheduling, and hardware-aware deployment [29, 30].

4.1 Learning Paradigms and Communication Functions

Table 2 maps the main learning paradigms to the communication tasks they support and the deployment costs used to assess them. The categories overlap in practice: a deep model may be trained federatively, compressed for a gateway, and managed by an RL-based scheduler [31].

Labeled data support fault classification, activity recognition, traffic prediction, and link-quality estimation. Where labels are sparse, unsupervised and semi-supervised models identify clusters or anomalous behavior from local streams [32, 33]. The choice is governed as much by annotation cost and false-alarm tolerance as by predictive accuracy.

Deep networks are used for images, speech, and multivariate sensor series because they learn features directly from high-dimensional input. On edge hardware, that advantage is weighed against memory, FLOPs, and inference delay. Compressed CNN, RNN, or Transformer variants are typically deployed at gateways or higher-capacity edge servers rather than the smallest sensing nodes [30, 34].

Table 2: Mapping ML techniques to IoT communication functions

ML paradigm	Communication-system function	Edge deployment detail	Key trade-off / metric
Supervised learning	Fault classification, traffic prediction, link-quality estimation	Gateway or edge-server inference using labeled historical data	Accuracy versus labeling cost; inference latency
Unsupervised / semi-supervised learning	Anomaly detection, clustering of devices or traffic flows	Local pattern discovery when labels are scarce	False alarms versus detection sensitivity
Deep learning	Video analytics, multivariate time-series inference, semantic feature extraction	Compressed CNN/RNN/Transformer variants at edge nodes	Model size, FLOPs, memory footprint, accuracy
Reinforcement learning	Adaptive scheduling, routing, bandwidth allocation, offloading	Policies trained from local state: queue, channel, energy, deadline	Convergence stability, exploration risk, delay/energy reward
Federated learning	Collaborative model training without raw-data upload	Client selection, update compression, secure/asynchronous aggregation	Communication rounds, privacy leakage, non-IID robustness
TinyML / knowledge distillation	Extreme-edge filtering and lightweight local inference	Quantized student models on MCUs or low-power gateways	Energy per inference, compression ratio, accuracy loss

Reinforcement learning is applied when communication actions must be revised as the environment changes. Queue length, channel state, residual energy, and task deadlines form the state; scheduling, routing, bandwidth assignment, and offloading become actions [35, 36]. Results depend on reward design, exploration policy, and whether the learned policy remains stable outside its training conditions.

Federated learning keeps raw data on participating devices, but repeated parameter or gradient exchange can consume more bandwidth than the original application traffic. Non-IID data, intermittent links, and slow clients further delay convergence. Client selection, update compression, asynchronous aggregation, secure aggregation, and radio-aware scheduling are therefore part of the FL system design rather than optional additions [37, 38].

4.2 Lightweight Models and Edge Deployment

Resource-limited devices cannot run every model used in a cloud experiment. Edge deployment usually begins by reducing parameter count, numerical precision, memory movement, or the frequency of communication [29, 39].

Pruning removes weights or structures that contribute little to the output. Quantization uses lower-precision arithmetic, low-rank factorization reduces matrix cost, and knowledge distillation trains a compact student from a larger teacher [40]. Each method changes the balance among accuracy, model size, memory footprint, and execution time.

A smaller model can lower inference delay and energy consumption, but compression ratios alone do not predict device performance. Hardware support, memory access, batch size, and operator availability affect the actual gain. Reported evaluations therefore need model size and FLOPs alongside measured latency, energy per inference, and task accuracy [41].

TinyML pushes this process to microcontrollers with kilobyte-scale memory and milliwatt-scale power budgets [42, 43]. A typical workflow combines hardware-aware training, conversion, quantization, and an

optimized runtime. Common tasks include vibration monitoring, keyword spotting, gesture recognition, and simple anomaly detection.

Running these tasks at the sensing node changes communication behavior. The device can discard routine samples and send an event label or compact feature vector, which reduces radio use and cloud traffic [44]. The savings must be compared with the extra flash, RAM, and energy consumed by continuous inference.

Deployment decisions are therefore hardware- and task-specific. Federated or distributed learning can update lightweight models across devices, but the update schedule and model format must fit the same resource budget as local inference [38, 45].

4.3 Data Distribution and Privacy Protection

EI data are split across devices, organizations, and network tiers. Local datasets are often non-IID, imbalanced, and time-varying, while raw signals from healthcare, industry, and vehicles may be too sensitive or costly to centralize. Federated learning coordinates training without moving those raw records [25, 26], but its communication and privacy properties depend on the rest of the protocol.

Model updates can expose training examples through gradient inversion or membership inference, and compromised clients can poison an aggregate. Heterogeneous hardware and unreliable links make these attacks harder to distinguish from benign variation. Padmavathi et al.'s AI-SET framework combines edge intrusion detection, privacy-preserving FL, and trust-based access control, illustrating how participant screening and learning can be implemented together [47].

Differential privacy limits the influence of individual samples by adding calibrated noise to gradients, parameters, or outputs. In small or highly non-IID local datasets, stronger noise can slow convergence and reduce task accuracy. Privacy budgets therefore have to be selected together with batch size, communication rounds, and application error tolerance [34, 36].

Blockchain-based coordination records model-update provenance, aggregation events, reputation values, or access decisions in a shared ledger. Zhang's 6G edge-network scheme combines blockchain, encryption, and threshold sharing to reduce dependence on a single aggregator [46]. The added consensus delay, storage, and protocol complexity may be unacceptable for short-deadline tasks.

Operational systems often combine FL, secure aggregation or encryption, differential privacy, and trust management. The remaining design question is how much protection a task can afford before security overhead removes the latency or bandwidth benefit of edge processing [26, 37, 46].

4.4 Performance Metrics

End-to-end latency covers sensing, transmission, queueing, scheduling, inference, and response. Distributed training adds model-update and synchronization delay [12, 20]. Deadline satisfaction and tail latency are useful complements to an average, especially in control loops where a small number of late decisions can dominate system risk.

Energy measurements should distinguish computation from radio use. Useful indicators include joules per inference, energy per transmitted bit, joules per completed task, and the energy-delay product. They reveal cases in which a lower latency is purchased through higher power draw or more frequent communication [29, 45, 48].

Predictive quality is reported through accuracy, precision, recall, F1-score, mean absolute error, or AUC. Edge studies also need convergence speed, non-IID robustness, and the accuracy lost after pruning or quantization. Privacy and security evaluations may add privacy budget, attack success rate, and resistance to poisoned updates [25, 36, 47].

Communication cost can be measured as uplink and downlink volume, synchronization rounds, compression ratio, packet delivery ratio, or spectrum efficiency. Knowledge-distillation-based EI, for example, exchanges less information than full-model synchronization when a compact student is sufficient [48]. A credible evaluation reports communication, learning quality, latency, and energy on the same experimental setup; optimizing one measure in isolation gives an incomplete result.

5. Applications of Edge Intelligence in IoT

5.1 Urban and Transportation Networks

Urban EI deployments place models at intersections, roadside units, buildings, and utility gateways. Camera and sensor data can be fused locally for traffic-light control, congestion alerts, parking guidance, environmental anomaly detection, or emergency response [6, 15]. Filtering at the source also limits the volume sent to city platforms, while digital-twin systems use selected updates to keep virtual models aligned with physical infrastructure [49].

Vehicular networks add mobility and rapidly changing wireless links. Roadside and onboard processors handle cooperative perception, handover support, traffic prediction, and communication-aware offloading under tight deadlines [17, 18, 19]. Local fusion is useful when safety messages must be prioritized, but its reliability depends on channel quality, vehicle density, and the freshness of shared observations.

5.2 Industrial IoT

Industrial EI is tied directly to machines and control loops. Vibration, acoustic, temperature, and current signals can be analyzed near the equipment to flag an incipient fault before a cloud round trip completes [15, 45]. Local processing also reduces backhaul traffic and keeps alarms available during intermittent external connectivity.

Digital twins extend this arrangement by linking machine telemetry with a continuously updated operational model. Edge nodes preprocess measurements, synchronize selected state changes, and run local models for quality inspection, scheduling, or process adjustment. Attaran et al. describe IIoT as the data layer for industrial twins and AI as the analytical layer used for simulation and decision support [49].

5.3 Healthcare

Healthcare systems cannot treat latency and privacy as separate requirements. Wearables, bedside monitors, imaging devices, and ambient sensors generate streams that may require immediate interpretation but contain sensitive patient information. Edge models can detect an abnormal physiological pattern or produce a preliminary result near the point of care, reducing response time and the amount of raw data leaving the clinical setting [16].

Cross-institution training introduces a different problem: hospitals hold useful but non-identical datasets and often cannot pool them. Federated learning supports collaborative model updates, while encryption, differential privacy, and access control reduce exposure during exchange [25, 47]. Communication asymmetry, non-IID clinical data, and leakage from model updates remain practical limits.

5.4 Agriculture and Remote Monitoring

Farms and environmental sites often have intermittent backhaul and widely dispersed sensors. Edge nodes can combine soil moisture, temperature, humidity, and crop images to schedule irrigation, detect disease, control a greenhouse, or monitor livestock without maintaining a continuous cloud connection [50]. The local decision path is especially useful when a radio link is slow, costly, or unavailable.

These systems also mix sensing modalities and time scales. A gateway may trigger irrigation from soil and weather readings, flag an abnormal microclimate, or schedule a drone inspection after a visual model detects crop stress. Lightweight inference and task-aware filtering reduce routine transmissions, although model updates and power supply remain constraints in remote deployments [50].

6. Challenges and Future Directions

6.1 Resource Constraints and Task Offloading

Edge nodes share limited processors, memory, storage, and battery capacity with sensing and communication tasks. Even a modest model can interfere with packet handling or control deadlines. Offloading

is therefore modeled across device, gateway, edge-server, and cloud tiers rather than as a single device-versus-cloud choice [12, 45].

A policy must account for task size, model stage, queue state, radio condition, deadline, server load, and residual energy. Partial offloading or model partitioning is often preferable because feature extraction and final decision stages have different compute and latency profiles. Distilled student models provide another option: a small device model handles routine cases while a larger edge or cloud model supports updates or difficult inputs [48].

6.2 Security and Privacy

EI inherits spoofing, denial-of-service, and data-tampering threats from IoT networks and adds attacks against models and training pipelines. Adversarial examples, poisoned updates, backdoors, and inference attacks can enter through many heterogeneous participants, making the attack surface broader than in a centralized service [37, 43].

Secure aggregation, differential privacy, trusted execution, access control, and blockchain audit logs cover different parts of this surface [46, 47]. All consume communication, computation, or storage. Protection therefore has to reflect task criticality, the trust assigned to each participant, and the delay budget; a uniform security stack may be too costly for low-power nodes.

6.3 Communication Bottlenecks

Moving intelligence to the edge does not remove the network bottleneck. Dense access, interference, fluctuating link quality, and asymmetric uplink capacity can dominate distributed training and collaborative inference. Too many synchronization rounds erase latency savings, while stale or missing updates reduce convergence quality and fairness [21, 27].

Current work reduces this cost through compressed updates, asynchronous aggregation, selective client participation, semantic filtering, and learning-aware scheduling. These methods decide what information to send, at what fidelity, and under which network conditions [23, 51]. Their performance should be tested under packet loss, changing client availability, and realistic radio contention rather than an ideal transport layer.

6.4 Model Adaptability

Workloads, user behavior, wireless conditions, and sensor distributions change after deployment. A model trained at one site or season may lose accuracy when moved to another. Healthcare, transportation, and agriculture make this problem visible because local context differs across devices and regions [15, 20].

Transfer learning, continual learning, and meta-learning reduce the amount of retraining needed for a new client or task. Federated meta-learning seeks an initialization that adapts with a small number of local updates. The study in [51] reports that this combination can reduce communication cost while improving classification in edge IoT networks, linking model adaptation directly to update efficiency.

6.5 Energy Efficiency

Energy use determines device lifetime, maintenance intervals, and the operating cost of large edge deployments. It includes local compute, memory access, radio transmission, and repeated model updates. A pipeline that shifts traffic away from the cloud can still consume more total energy if devices run an oversized model continuously [29, 30].

TinyML, quantization, pruning, hardware-aware architecture search, duty cycling, and selective communication reduce different parts of this budget [40, 48]. Comparisons should report energy per inference and energy-delay product together with accuracy and latency. Without measurements on target hardware, claims of greener edge intelligence remain difficult to assess.

6.6 Research Priorities

6G-oriented EI is increasingly linked with semantic communication and reconfigurable intelligent surfaces, where the radio environment, the meaning carried by a transmission, and the learning objective are optimized

together [52, 53]. The open issue is how to validate these schemes when channel models, application semantics, and edge hardware vary across testbeds.

Cross-layer orchestration must also coordinate sensing, radio access, offloading, model updates, security, and control under changing workloads. Lightweight foundation models and tiny federated learning may broaden the tasks handled near the data source, but compression, uncertainty estimation, privacy, and energy use require joint evaluation. Shared benchmarks and open testbeds are needed to compare these choices with the same communication and learning metrics [52, 53].

7. Conclusion

The reviewed evidence points to a consistent pattern: edge intelligence is effective when the learning model and the communication path are designed together. Supervised and deep models support local perception, RL policies adapt scheduling and offloading, and federated learning coordinates sites that cannot centralize raw data. Each method introduces a deployment cost in memory, radio traffic, energy, convergence time, or privacy risk.

Hardware and application context determine which trade-off is acceptable. Transportation studies emphasize deadlines and link variability; healthcare adds data governance; agriculture faces intermittent connectivity and power limits; industrial systems require reliable control and fault response. Lightweight models, selective communication, and privacy controls are therefore system choices rather than universal add-ons.

The next research step is empirical. Methods should be compared on shared edge platforms under non-IID data, packet loss, changing client participation, and measured power consumption. Such experiments would show whether a gain in accuracy or average latency survives the communication, security, and energy conditions of a real deployment.

References

[1] Mrabet, H., Belguith, S., Alhomoud, A., & Jemai, A. (2020). A survey of IoT security based on a layered architecture of sensing and data analysis. Sensors, 20(13), 3625. https://www.mdpi.com/1424-8220/20/13/3625

[2] Paul, B. (2022). Internet of Things (IoT), three-layer architecture, security issues and countermeasures. Springer. https://link.springer.com/chapter/10.1007/978-981-16-5655-2_3

[3] Rashid, T., & Mustafa, S. (2023). A review on IoT: Layered architecture, security issues and protocols. Journal of Computer Science & Network Security. https://koreascience.kr/article/JAKO202329158379658.page

[4] El Hanine, M., & El-Yahyaoui, A. (2024). Three Layer IoT Architecture: Attacks and Security Mechanisms. IEEE. https://ieeexplore.ieee.org/document/10742994

[5] Kaur, H., & Kumar, R. (2020). A survey on Internet of Things (IoT): Layer-specific and domain-specific architectures. Springer.

[6] Wang, X., Han, Y., Leung, V. C. M., et al. (2020). Convergence of edge computing and deep learning: A comprehensive survey. IEEE Communications Surveys & Tutorials. https://ieeexplore.ieee.org/document/8976180

[7] Xu, D., Li, T., Li, Y., Su, X., & Tarkoma, S. (2020). A survey on edge intelligence. IEEE Communications Surveys & Tutorials (arXiv version).

[8] Duan, S., Wang, D., Ren, J., et al. (2022). Distributed artificial intelligence empowered by end-edge-cloud computing. IEEE Communications Surveys & Tutorials. https://ieeexplore.ieee.org/document/9933792

[9] Baccour, E., Mhaisen, N., Abdellatif, A. A., et al. (2022). Pervasive AI for IoT applications: A survey. IEEE Communications Surveys & Tutorials. https://ieeexplore.ieee.org/document/9866918

[10] Gill, S. S., Golec, M., Hu, J., et al. (2025). Edge AI: A taxonomy, systematic review and future directions. Cluster Computing. https://link.springer.com/article/10.1007/s10586-024-04686-y

[11] Tang, F., Mao, B., Kawamoto, Y., Kato, N., & Shen, X. (2021). Machine learning for intelligent end-to-end communication toward 6G. IEEE Communications Surveys & Tutorials. https://ieeexplore.ieee.org/document/9403380

[12] Luo, Q., Hu, S., Li, C., et al. (2021). Resource scheduling in edge computing: A survey. IEEE Communications Surveys & Tutorials. https://ieeexplore.ieee.org/document/9519636

[13] Chang, Z., Liu, S., Xiong, X., & Cai, Z. (2021). Edge-computing-powered artificial intelligence of things. IEEE Internet of Things Journal. https://ieeexplore.ieee.org/document/9453402

[14] Wu, Y., Dai, H. N., Wang, H., & Xiong, Z. (2022). A survey of intelligent network slicing for industrial IoT. IEEE Communications Surveys & Tutorials.

[15] Bourechak, A., Zedadra, O., Kouahla, M. N., & Guerrieri, A. (2023). At the confluence of AI and edge computing in IoT-based applications. Sensors, 23(3), 1639.

[16] Amin, S. U., & Hossain, M. S. (2020). Edge intelligence and Internet of Things in healthcare: A survey. IEEE Access. https://ieeexplore.ieee.org/document/9294145

[17] Gong, T., Zhu, L., Yu, F. R., & Tang, T. (2023). Edge intelligence in intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems.

[18] Xu, X., Li, H., Xu, W., Liu, Z., & Yao, L. (2021). Artificial intelligence for edge service optimization in IoV. Tsinghua Science and Technology.

[19] Yan, G., Liu, K., Liu, C., & Zhang, J. (2024). Edge intelligence for Internet of Vehicles: A survey. IEEE Transactions on Consumer Electronics.

[20]Merenda, M., Porcaro, C., & Iero, D. (2020). Edge machine learning for AI-enabled IoT devices: A review. Sensors, 20(9), 2533.

[21] Zhang, J., Qu, Z., Chen, C., et al. (2021). Edge learning: The enabling technology for distributed analytics. ACM Computing Surveys.

[22] Filho, C. P., Marques, E., Chang, V., & dos Santos, L. (2022). Distributed machine learning in edge computing: A review. Sensors, 22(7), 2665.

[23] Campolo, C., Iera, A., & Molinaro, A. (2023). Network for distributed intelligence: A survey. IEEE Access.

[24] Tang, S., Cui, M., Qi, L., & Xu, X. (2023). Edge intelligence with distributed processing of DNNs.

[25] Ficco, M., Guerriero, A., Milite, E., & Palmieri, F. (2024). Federated learning for IoT devices. Information Sciences.

[26] Ramalingam, V., et al. (2026). Hybrid federated learning for privacy-preserving IoT. Scientific Reports.

[27] Cao, S., et al. (2026). KG-AsyncFed: Asynchronous federated continual learning. Computer Networks.

[28] Han, S., Mao, H., & Dally, W. J. (2021). Deep compression: Compressing deep neural networks. IEEE TPAMI.

[29] Rajapakse, V., et al. (2023). Intelligence at the extreme edge: A survey on TinyML. ACM Computing Surveys.

[30] Somvanshi, S., Islam, M. M., & Chhetri, G. (2025). From TinyML to TinyDL: A survey. ACM Computing Surveys.

[31] Arif, M., & Rashid, M. (2025). Model conversion and TinyML deployment. Computers, Materials & Continua.

[32] Fragkou, E., & Katsaros, D. (2024). Decentralized federated learning and TinyML. Future Internet.

[33] da Silva, C. N., & Prazeres, C. V. S. (2025). Tiny federated learning for constrained sensors. IEEE Sensors Reviews.

[34] Kalimuthu, N. (2026). Privacy-preserving technologies in edge analytics.

[35] Zhao, Y., et al. (2026). Explainable edge intelligence with split federated learning. IEEE Transactions on Consumer Electronics.

[36] Tang, R. R. (2025). Federated anomaly detection with differential privacy. Informatica.

[37] Farhan, M., et al. (2026). Emerging technologies in IoT security. Elsevier.

[38] Jain, V., et al. (2026). TinyFed6G: Federated learning with TinyML. IEEE Wireless Communications.

[39] Wu, Y., et al. (2023). Intelligent network slicing management.

[40] Essahraui, S., & Lamaakal, I. (2026). TinyML-based edge intelligence for IoT. IEEE IoT Journal.

[41] Pazmiño Ortiz, L. A., & Maldonado Soliz, I. F. (2025). Advancing TinyML in IoT. Future Internet.

[42] Joshua, R. G., et al. (2026). TinyML for anomaly detection.

[43] Baduwal, M., et al. (2026). Federated learning: Core challenges and opportunities.

[44] Xu, Z., et al. (2024). Edge intelligence for real-time decision making.

[45] Walia, G. K., et al. (2023). AI-empowered edge resource management for IoT.

[46] Zhang, F. (2026). A blockchain-enabled privacy-preserving and auditable federated learning scheme for 6G edge intelligent networks. Journal on Wireless Communications and Networking. https://doi.org/10.1186/s13638-025-02558-6

[47] Padmavathi, V. (2025). A federated edge intelligence framework with trust based access control for secure and privacy preserving IoT systems. Scientific Reports. https://doi.org/10.1038/s41598-025-19712-1

[48] Cui, Z., Pozi, M. S. B. M., & Mohsin, M. F. B. M. (2026). Latency-efficient edge intelligence in IoT networks using knowledge distillation. Discover Computing, 29, 199. https://doi.org/10.1007/s10791-026-10080-6

[49] Attaran, S., Attaran, M., & Celik, B. G. (2024). Digital Twins and Industrial Internet of Things: Uncovering operational intelligence in industry 4.0. Decision Analytics Journal, 10, 100398. https://doi.org/10.1016/j.dajour.2024.100398

[50] Li, L. (2026). Artificial intelligence of things for smart agriculture with edge computing and large language models. Computers and Electronics in Agriculture.

[51] Xiaofeng, Z., Qiaosong, F., Jiaqiang, P., & Yuwen, Q. (2025). A federated meta-learning aided intelligent edge framework by using the parameter optimization approach. Journal on Wireless Communications and Networking. https://doi.org/10.1186/s13638-025-02511-7

[52] Ahmed, M., Raza, S., Soofi, A. A., Khan, F., Khan, W. U., Xu, F., Chatzinotas, S., Dobre, O. A., & Han, Z. (2024). A survey on reconfigurable intelligent surfaces assisted multi-access edge computing networks: State of the art and future challenges. Computer Science Review, 54, 100668. https://doi.org/10.1016/j.cosrev.2024.100668

[53] Ogenyi, F. C., Ugwu, C. N., & Ugwu, O. P.-C. (2025). A comprehensive review of AI-native 6G: integrating semantic communications, reconfigurable intelligent surfaces, and edge intelligence for next-generation connectivity. Frontiers in Communications and Networks, 6, 1655410. https://doi.org/10.3389/frcmn.2025.1655410

Funding

This research received no external funding.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgment

This paper is an output of the science project.