

Reinforcement Learning for Energy-Constrained Edge IoT in Smart Cities: A Systematic Review and Research Roadmap

Dan Dan-Abia^{*1}, Kingsley Udofia², Dr. Kufre Udofia³

^{*1}Department of Electrical/Electronics Engineering, University of Uyo, Uyo, Nigeria

²Department of Electrical/Electronics Engineering, University of Uyo, Uyo, Nigeria

³Department of Electrical/Electronics Engineering, University of Uyo, Uyo, Nigeria

Corresponding Author: Dan Dan-Abia

Abstract

Smart city infrastructure relies on battery-backed, energy-harvesting edge IoT nodes for applications such as street lighting, environmental sensing, and traffic monitoring. Managing energy on such nodes is a sequential decision problem under stochastic harvesting, time-varying demand, and strict hardware limits. Reinforcement learning (RL) has become a promising alternative to hand-coded and common model-based power management systems, yet the literature remains fragmented across sensor networks, communications, energy systems, and embedded machine learning. This paper presents a structured review of RL for energy-constrained edge IoT control in smart cities. A review protocol is defined, a taxonomy of RL methods organized by edge deployability is proposed, and applications are synthesized across energy-harvesting power management, smart lighting, traffic and environmental sensing, and communication control. A layered reference architecture for on-device RL control is then proposed, combining compact feature engineering, resource-matched policies, a safety supervisor, and a fleet-level learning loop, and is instantiated for a solar street-lighting node. Guidance is given on environment modeling, training methodology, hardware deployment, and security, and RL is compared with non-RL control. Finally, open challenges are identified, including sample efficiency, nonstationarity, on-device learning cost, safety, and benchmarking, and a prioritized research roadmap is outlined.

Keywords: Adaptive control, edge computing, energy harvesting, Internet of Things, reinforcement learning, smart cities, systematic review, TinyML

Date of Submission: 04-10-2026

Date of acceptance: 14-10-2026

I. INTRODUCTION

Smart cities depend on large numbers of networked devices responsible for sensing, deciding, and acting at the network edge. These include adaptive street lights, air quality monitors, traffic sensors, and waste-bin monitors [1], [2]. Many of these nodes are deployed where grid power or regular battery replacement is impractical, so they are commonly paired with a battery and a photovoltaic (PV) harvester [3]. Energy harvesting turns power management into a sequential decision problem: harvested energy is stochastic, service demand varies over the day, the battery is finite, and the design goal is energy-neutral operation, in which long-run consumption does not exceed long-run harvest [4].

Typical solutions include adaptive duty cycling [5], energy-aware scheduling [6], and threshold or rule-based controllers built on harvest predictors [4]. These methods work well when the environment is well modeled, but they typically require per-site tuning and degrade under distribution shift such as seasonal change, prolonged cloud cover, or component ageing. Reinforcement learning (RL) instead learns a control policy from interaction with the environment [7], and the success of deep RL [8], [9] has stimulated broad interest in networking [10], [11] and edge intelligence [12], [13]. However, energy-constrained edge nodes pose a distinctive challenge: the learning agent itself consumes energy, must fit within microcontroller memory and compute budgets [14], [15], and must never compromise service or battery safety while it learns.

Existing surveys treat deep learning at the edge [12], [13] or RL in communications [10] in general terms, and rarely examine RL specifically for energy-limited, harvesting-powered control nodes in urban infrastructure. This paper fills that gap. Our contributions are: (i) a clear review protocol adapted from PRISMA [16] and the Kitchenham guidelines [17]; (ii) a taxonomy of RL methods arranged by suitability for resource-limited edge locations, and a synthesis of applications and of state, action, and reward design patterns across smart-city domains; (iii) a practical guide to environment modeling, training methodology, hardware deployment, and security and trust; (iv) a layered reference architecture for on-device RL control with a safety supervisor,

instantiated for a solar street-lighting node; and (v) a prioritized research roadmap and a recommended evaluation protocol.

II. BACKGROUND

2.1 Energy-Constrained Edge IoT and Energy Harvesting

A typical edge node is made up of a sensing front end, a microcontroller (MCU), a radio, an actuator (for example, a dimmable LED driver), an energy store, and an energy harvester. MCUs in this class, such as Arm Cortex-M devices, offer tens to hundreds of kilobytes of RAM and no floating-point acceleration in the lowest tiers [15], [18]. Energy-neutral operation requires balancing consumption against a harvest that is diurnal, seasonal, and weather-dependent [3], [4]. For solar street lighting the difficulty is structural: the load peaks at night when harvest is zero, so the battery must buffer the day's energy, and multi-day low-irradiance spells can cause depletion and loss of service.

2.2 Reinforcement Learning Formulation

The control problem is modeled as a Markov decision process (MDP) defined by states S , actions A , transition dynamics P , reward r , and discount factor γ [7], [19]. The agent seeks a policy π that maximizes expected discounted return:

$$\pi^* = \operatorname{argmax}_{\pi} E[\sum_{t=0}^{\infty} \gamma^t r_t] \quad (1)$$

Typical state variables include battery state of charge, a harvest estimate or trend, time of day and season, and a demand proxy such as ambient light or traffic count. Actions include the duty cycle, dimming level, sampling rate, transmit power, or an offloading decision. A common reward structure trades service utility against depletion risk:

$$r_t = w_u u_t - w_d \mathbf{I}[E_t < E_{\min}] - w_o o_t \quad (2)$$

where u_t is delivered utility (for example, illumination adequacy), $\mathbf{I}$ penalizes battery depletion below a floor $E_{\min}$, and o_t penalizes energy overflow or waste. The tabular Q-learning update [20] is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha [r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (3)$$

2.3 Smart City Deployment Requirements

Urban edge deployments are expected to run unattended for many years, to tolerate intermittent connectivity, and to remain heterogeneous across vendors and sites. These requirements favor controllers that are lightweight, adapt to local conditions, degrade gracefully, and can be audited by operators.

III. ENERGY HARVESTING AND BATTERY MODELING FOR RL ENVIRONMENTS

Most RL studies in this domain are trained and evaluated in simulation, so the fidelity of the environment model largely determines whether reported gains are credible. This section focuses on the harvest, storage, and load models that an RL environment for a solar-powered node should include.

3.1 Solar Harvest Models

PV panel harvest power is modeled as:

$$P_{h,t} = \eta_{pv} \cdot A \cdot G_t \cdot (1 - \zeta_t) \quad (4)$$

where G_t is the solar irradiance, A the panel area, η_{pv} the conversion efficiency, and ζ_t a loss factor capturing soiling, shading, and temperature effects. Irradiance can be taken from measured traces or synthesized as a clear-sky envelope multiplied by a stochastic cloud attenuation process. Whichever is used, the environment should include multi-day low-irradiance spells and seasonal variation, since these are the events that drive battery depletion. Synthetic generators should be validated against measured data from the target region. Harvest prediction, when used as an input feature, is commonly performed with exponentially weighted moving averages [4] or weather-conditioned averages [21]; the forecast error should be simulated rather than assuming a perfect predictor.

3.2 Battery and Storage Models

The simplest storage model, in which the stored energy level evolves, is specified as:

$$E_{t+1} = \operatorname{clip}(E_t + \eta_c P_{h,t} \Delta t - P_{l,t} \Delta t / \eta_d - E_{\text{leak}}, 0, E_{\max}) \quad (5)$$

where $P_{l,t}$ is the load power, η_c and η_d are charge and discharge efficiencies, E_{leak} is self-discharge, and $E_{\max}$ is the usable capacity. More detailed models add voltage dynamics, temperature dependence, and capacity fade with cycling [22]. Ideal bucket models are convenient but tend to overstate feasibility in practice, so it is recommended to include at least conversion efficiencies, leakage, and a protective lower limit $E_{\min}$. Where battery lifetime is an objective, a cycle-depth or throughput-based ageing term should be added so that the agent is not rewarded for policies that deplete the battery deeply and often.

3.3 Load and Demand Models

The load in adaptive lighting is the LED power at each dimming level, and the demand for illumination depends on the ambient light and on the arrival of pedestrians or vehicles, often modeled as a stochastic arrival process. The load for sensing nodes is the sensing, processing, and radio energy per sample and per packet. The decision interval Δt is also a design choice: shorter intervals give finer control at the cost of more decisions, and thus more energy spent by the agent.

3.4 Calibration and Validation

Parameters should be set from datasheets and, where possible, bench measurements, and results should be checked for sensitivity to the most uncertain parameters. Explicit reporting of parameters, such as battery capacity and LED power levels, allows others to reproduce and compare results, as in the case study of Section VIII. The gap between simulation and deployment is narrowed by hardware-in-the-loop (HITL) validation, in which the learned policy runs on the target MCU embedded in the simulated environment.

IV. REVIEW METHODOLOGY

4.1 Research Questions

The review addresses five research questions. **RQ1:** Which RL families have been used to control energy-constrained edge nodes? **RQ2:** Which reward and state designs are used, and in which smart-city applications? **RQ3:** How is the computing, memory, and energy cost of the on-device RL agent considered? **RQ4:** How are solutions evaluated (simulation versus hardware, baselines, datasets)? **RQ5:** Which challenges remain unanswered?

4.2 Search Strategy

Searches were made through IEEE Xplore, ACM Digital Library, ScienceDirect, SpringerLink, Scopus, and Web of Science, with arXiv used only for preprints of published papers. The main search string combines three concept groups with AND: (“reinforcement learning” OR “Q-learning” OR “deep Q” OR “actor-critic”) AND (“energy harvesting” OR “energy-constrained” OR “battery” OR “solar”) AND (“IoT” OR “edge” OR “sensor node” OR “smart city” OR “street light”). The search window covers 2006 to 2026. A supplementary targeted web search for street-lighting, TinyML, and on-device RL studies was run on 12 October 2026 to capture recent work.

4.3 Selection and Data Extraction

Studies are included if they (I1) use RL for the control or management of an energy-limited edge device or node, (I2) are peer reviewed or archival, (I3) are written in English, and (I4) include an evaluation. Studies are excluded if they are performed only in the cloud with no energy dimension, or are extended abstracts with no results. Screening is done in two stages (title/abstract, then full text), and disagreements are resolved through discussion.

V. TAXONOMY OF RL APPROACHES FOR EDGE CONTROL

We classify RL methods according to their suitability for resource-constrained edge hardware (Table 1).

Table 1: RL Families and Their Edge Suitability

Family	Typical methods	Edge fit	Main limitation
Tabular value-based	Q-learning [20], SARSA	Minimal memory and compute; runs on low-end MCUs	State explosion; needs discretization
Linear function approx.	Semi-gradient Q/SARSA with tile coding or hand-crafted features [7]	Low cost; generalizes across nearby states	Depends on feature quality
Deep value-based	DQN [8], Double DQN [23], Dueling [24]	Handles rich state; strong performance	Memory and compute; needs compression
Policy gradient / actor-critic	DDPG [25], PPO [26], SAC [27]	Continuous actions (e.g., dimming level)	Sample-inefficient; tuning-sensitive
Multi-agent RL	Independent learners; centralized training [28]	City-scale coordination	Non-stationarity; communication cost
Safe / constrained RL	CPO [29], shielding [30]	Enforces energy and safety limits	Added design complexity
Offline / model-based	CQL [31]; learned simulators [32]	Uses logged data; less online risk	Distribution shift

5.1 Tabular and Linear Methods

Tabular Q-learning and SARSA remain the most widely deployed RL methods in harvesting-powered nodes because the learned table is small and inference is a lookup [33], [34]. Their weakness is discretization: fine grids of battery level, time, and weather quickly become large, and coarse grids lose information. Linear function approximation over engineered features relaxes this trade-off by sharing learning across similar states while keeping inference to a dot product [7].

5.2 Deep Value-Based Methods

Deep Q-networks approximate the action-value function with a neural network and use experience replay and target networks for stability [8], with Double and Dueling variants reducing overestimation and improving learning efficiency [23], [24]. They accommodate high-dimensional states, such as forecast vectors, but the network must be quantized and pruned to fit MCU memory [35]–[37], and runtime kernels such as CMSIS-NN and TensorFlow Lite Micro are commonly used [14], [18].

VI. APPLICATIONS IN SMART CITY EDGE SYSTEMS

Table 2 summarizes representative works spanning energy-harvesting power management and adjacent smart-city control problems whose formulations transfer to energy-constrained edge nodes. The table is a representative sample rather than an exhaustive count, and the synthesis that follows is therefore qualitative.

Table 2: Representative Studies by Application Domain

Ref.	Domain	Method	Objective
[33]	EH wireless sensor node	RL-based time-of-day power provisioning	Sustainable node operation
[38]	EH wireless sensor network	Actor–critic energy manager (RLMan)	Energy-neutral operation
[34]	Solar EH sensor node	RL power management	Utility within energy budget
[41]	EH sensor configuration	RL-based automatic configuration (ACES)	Sensing quality and uptime
[65]	Indoor solar EH sensors	RL sampling-rate configuration with shared policies	Maximize sensed data without depleting storage
[66]	EH wearable IoT device	Lightweight RL energy manager (tinyMAN) for MCUs	Device utilization under harvesting and battery limits
[67]	Street lighting (non-RL)	Deep-learning (YOLOv5) intensity control on embedded GPU	Reduce late-night energy use
[42]	Traffic signal control	Deep Q-learning	Reduce vehicle delay
[39]	Networked traffic control	Multi-agent deep RL	Coordinated control
[43]	Building energy	Deep RL	Cost and comfort
[44]	Wireless multiple access	Deep RL (DLMA)	Throughput and fairness
[45]	Cluster resource management	Policy gradient (DeepRM)	Job slowdown

6.1 Energy-Harvesting Power Management

What matters is not the sophistication of the algorithm but whether the policy can be stored and evaluated within the node’s memory and energy budget. The most direct application is using RL to set the duty cycle or activity level of harvesting nodes to maintain battery health and maximize service utility [33], [34], [38], [41]. Recent work extends this line in two practical directions: reducing training time and the number of policies needed in large deployments by sharing one policy among nodes with similar conditions [65], and fitting the energy manager onto low-end microcontrollers [66]. The results of these studies tend to be positive compared with static or heuristic policies, but evaluation practice varies considerably: harvest traces, battery models, and baselines differ, which makes direct comparison difficult.

6.2 Smart Street Lighting and Environmental Sensing

Adaptive lighting is a natural fit for RL: the action (dimming level) is discrete or continuous, the demand depends on ambient light and pedestrian or vehicle traffic, and solar harvesting couples the daytime decision context to nighttime service [1]. The same applies to environmental monitoring, where the actions are the sampling and transmission rates and the utility is the freshness of the data. Although these applications are often cited as motivation, there are comparatively few rigorous evaluations of RL on harvest-powered nodes. Recent street-lighting work illustrates this gap: an embedded deep-learning controller on a Jetson Nano adjusts streetlight intensity from video-based detection to cut late-night energy use [67], but it is a supervised perception pipeline on a mains-class platform rather than a harvest-aware RL policy. Harvest-aware, battery-constrained RL for street lighting thus remains largely open.

6.3 Traffic and Urban Control

RL has been widely studied for traffic-signal control [39], [42] and for building energy management [43], [46]. While these systems are typically mains powered, their state and reward design, multi-agent structure, and demand-response formulations provide transferable patterns for battery-backed urban nodes.

6.4 Communication and Offloading Control

RL has also been applied to managing access to channels and resources in wireless networks [10], [44] and to scheduling compute jobs [45]. For energy-limited nodes, the cost of communication itself and the cost of the decision must both be taken into account, and the latter is not always modeled explicitly.

6.5 State, Action, and Reward Design Patterns

Three design patterns recur across the surveyed domains. First, state design is a compromise between information and size: battery state of charge and a cyclic time-of-day encoding are almost universally included, while harvest trends or demand proxies are included where the policy class can afford them [4], [33]. Second, action spaces tend to be small and discrete (a few duty-cycle or dimming levels), which suits value-based methods, while continuous action spaces call for actor-critic designs [25], [38]. Third, reward design includes a utility term along with penalties for depletion and waste; the weights of these penalties critically affect the policy, and a badly scaled penalty leads either to overly conservative policies that waste harvested energy or to aggressive policies that risk blackout [40], [47]. Reward weights should therefore be reported and, where possible, sensitivity tested.

VII. LEARNING PIPELINE AND TRAINING METHODOLOGY

This section outlines a suggested training strategy for RL controllers on energy-constrained nodes, derived from best practice in the RL community [47], [48] and adapted to the edge deployment scenario.

7.1 Environment Interface and Episode Design

The simulated node should provide a familiar reset/step interface, such as OpenAI Gym [49], so that tabular, linear, and deep agents can be compared on the same environment. An episode should span one or more days to weeks; the initial battery level, season, and weather regime should be randomized to prevent the agent from overfitting to a single trajectory at reset. The seeds for the environment and the agent should be recorded.

7.2 Training Procedures

Tabular and linear agents are typically trained using an ϵ -greedy exploration strategy and decaying learning rates, with feature scaling or tile coding for linear approximators [7]. Deep value-based agents use experience replay and a slowly updated target network [8], optimizing the temporal-difference loss:

$$L(\theta) = E[(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2] \quad (6)$$

where θ^- denotes the fixed target-network parameters. Common additions are Double Q-learning to reduce overestimation [23], prioritized replay for learning from informative transitions [50], and the Adam optimizer [51]. The time of day is strongly periodic, so encoding it as $\sin(\text{time of day})$ and $\cos(\text{time of day})$ can significantly improve learning speed.

7.3 Reward Design and Shaping

Reward weights determine the trade-off between service and energy conservation, and poor weights can produce degenerate behavior, such as an agent that holds the light at its lowest setting to avoid depletion penalties and thereby delivers almost no service. Learning can be accelerated without changing the optimal policy by using potential-based shaping [52]. If a minimum service level is mandatory, it is best imposed as a constraint or through a safety supervisor [29], [30] rather than a large penalty, which can destabilize learning. We recommend reporting the weights used and testing sensitivity to them.

7.4 Generalization and Robustness

Training should cover as many conditions as possible. Policies should be trained on diverse conditions (irradiance profiles, battery parameters, and demand patterns) using domain randomization [53], [54] and evaluated on held-out conditions. Leakage caused by the strong temporal correlation of weather is avoided by splitting data by season or by site. Stress tests (consecutive cloudy days, sensor dropout, battery capacity fade) expose failure modes that average-case evaluation would not reveal.

7.5 Statistical Reporting

Deep RL results can be sensitive to seeds and hyperparameters [48], so algorithms should be run at least 10 times with different seeds, with mean results reported together with confidence intervals, and all hyperparameters and the search budget of baselines documented alongside the proposed method. Ablations, such as removing individual state features, show what matters in the design and enable a trade-off analysis between feature richness and footprint.

7.6 Training for Compact Policies

A large policy can be distilled into a smaller student policy that is cheaper to run [55], and quantization-aware training can maintain accuracy at lower precision [36]. This train-large, deploy-small paradigm separates the node inference budget from the training budget and is integral to the hardware considerations outlined in Section IX.

VIII. REFERENCE ARCHITECTURE AND CASE STUDY

8.1 Layered Architecture

Following an analysis of the reviewed literature, we suggest a 5-layer structure for on-device RL-enabled energy-constrained edge nodes (Figure 1). In the first layer (energy-state estimation), the battery state of charge is monitored and the trend of energy harvest is estimated. A compact state vector is computed in Layer 2 (feature engineering), containing cyclic time encodings, harvest trend, forecast proxy, and demand proxy, which controls the size of the learning problem. In Layer 3 (policy), a policy class is selected according to the resource tier of the node: a tabular or linear policy for the smallest nodes and a quantized network for more capable ones. Layer 4 (safety supervisor) applies rule-based or shielded overrides that ensure a minimum service level and protect the battery floor regardless of the learned action [30]. Layer 5 (learning loop) pre-trains in simulation, fine-tunes or updates the model periodically, and can optionally use federated learning [56], [57] or offline RL [32] to aggregate fleet experience.

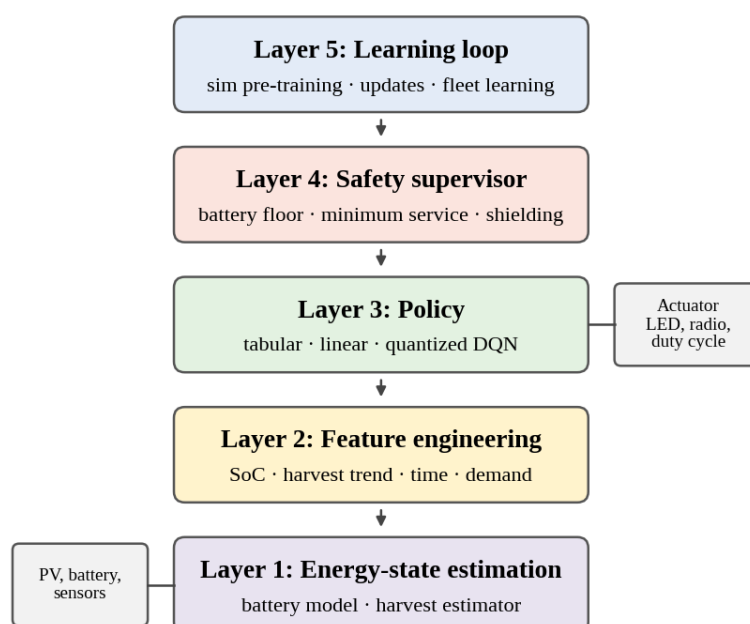


Figure 1: 5-layer reference architecture for on-device RL control of an energy-harvesting edge node.

Data flows from energy-state estimation through feature engineering to the policy, and the safety supervisor gates any action passed to the actuator.

8.2 Case Study: Solar Street-Lighting Node

The architecture is instantiated for a solar-powered street-light node with a battery capacity of approximately 100 kJ and three LED dimming levels of 0.2 W, 0.8 W, and 1.6 W. Layer 2 builds a 12-feature state vector; Layer 3 is realized with three policy classes (tabular Q-learning, linear function approximation, and a DQN), which are compared against five baseline controllers, while Layer 4 enforces the battery floor. The simulation environment is implemented in Python using NumPy, following the modeling guidance of Section III and the evaluation protocol of Section VII. Quantitative results for this configuration are outside the scope of this review.

IX. HARDWARE AND TINYML IMPLEMENTATION

9.1 Resource Tiers and Feasible Policies

Edge nodes vary widely in computational power, and the resource tier of a node determines which policy classes are feasible on it: tabular and linear policies suit the smallest tiers, while quantized networks suit nodes with more memory and compute. The ranges are not necessarily actual product specifications; feasibility depends on the firmware footprint and on the memory requirements of the rest of the application [14], [15].

9.2 Energy Cost of Decisions

Learning a policy has an energy cost, and it is worth learning only if the benefits outweigh that cost. The energy used per decision can be broken down into:

$$E_{\text{dec}} = E_{\text{inf}} + E_{\text{upd}} + E_{\text{mem}} \quad (7)$$

representing inference, learning update, and memory access. In neural workloads, memory movement is a major factor compared with arithmetic operations [37], [59], so a smaller model footprint and compact parameters are advantageous. For a street light drawing, say, a watt, the energy expended by a decision every few minutes is unimportant; for a microwatt-class sensor node that sleeps most of the time, it might not be. The decision interval and policy class should therefore be selected together based on the node's power profile.

9.3 Compression and Runtimes

The memory and compute of neural policies are reduced via quantization to 8-bit or lower, pruning, and distillation [35], [36], [55]. Tabular and linear policies can be represented as fixed-point arrays in flash and evaluated with a handful of integer operations. Optimized kernels for Cortex-M devices are embedded in the inference libraries and interpreters. Lightweight RL energy managers designed for low-end microcontrollers [66] show that this is practical for harvesting-powered devices.

9.4 Deployment Pipeline

A staged pipeline is recommended: (i) train and select the policy in simulation; (ii) export the policy as compact firmware code or data; (iii) test the policy with HITL tests; (iv) run the policy in shadow mode, where the RL agent's recommended action is logged while a baseline controller retains control; (v) deploy the policy gradually on a subset of nodes; and (vi) monitor key metrics automatically with a fallback to the baseline. Over-the-air updates should be reversible and signed.

X. COMPARISON WITH NON-RL CONTROL APPROACHES

RL should be compared with good non-learning alternatives, not weak baselines. This section situates it among the key families of methods for managing energy (summarized in Table 3).

10.1 Rule-Based Controllers

Rule-based controllers, such as dimming based on state-of-charge (SoC) thresholds and time schedules, are simple, transparent, and economical, and remain the industry standard. The downside of their inflexibility is that thresholds set for a specific site and season are less than optimal elsewhere, and they cannot react to fluctuations in demand.

10.2 Prediction-Based Heuristics

These methods estimate future harvest and then modify the duty cycle to compensate [4], [5], [21]. They are flexible to changes and efficient on-device, but their performance depends crucially on prediction accuracy, and they optimize a hand-specified objective instead of learning a long-horizon trade-off from experience.

10.3 Model Predictive and Optimization-Based Control

Model predictive control (MPC) is an optimization approach that acts over a receding horizon using a model of the system and forecasts [60]. It is explicit about constraints and works well when the model and forecast are correct, but it requires an optimization problem to be solved online, which is a heavy burden on MCU-class nodes, and it is sensitive to model error. Harvest-prediction heuristics do not guarantee real-time deadlines, whereas scheduling-based approaches such as lazy scheduling can [6]. A promising hybrid is the distillation of compact policies [55], in which MPC and optimization solutions computed at a gateway act as expert demonstrations.

10.4 When is RL the Right Tool?

RL is most appropriate when the dynamics are uncertain or non-stationary, no accurate analytic model is available, decisions have delayed consequences (for example, a dimmed light affects service for many hours), and a faithful simulator or logged data is available. RL is a poor choice if simple rules can meet the specification, if certifiable deterministic behavior is required, or if there is no simulator or safe way to explore. A pragmatic compromise is a learned policy that suggests actions, combined with a rule-based supervisor that guarantees the safety envelope. Table 3 compares the control approaches.

Table 3: Comparison of Control Approaches for Energy-Harvesting Nodes

Approach	Needs model?	Adaptivity	Compute	Main weakness
Rule / threshold	No	Low	Very low	Fixed; needs per-site tuning
Prediction-based heuristic [4]	Harvest predictor	Medium	Low	Sensitive to forecast error
MPC [60]	Yes	Medium (re-plans)	High	Model error; heavy online solve
Lazy scheduling [6]	Yes	Low to medium	Low	Task-model assumptions
RL (tabular/linear)	Simulator for training	High	Very low	Training effort; safety
RL (deep)	Simulator for training	High	Medium	Footprint; data hunger

XI. ENSURING SECURITY, PRIVACY, AND TRUST

Networked infrastructure nodes are difficult to protect when left unattended, and connecting them to learning systems creates new attack surfaces in addition to common IoT security issues [61].

11.1 Threat Model

Threats of particular concern are false data and erroneous sensor inputs (such as battery or light readings), firmware tampering, jamming or interception of communication, and poisoning of the data or model updates fed to a fleet learning system. The effects range from lost energy to blackouts at numerous nodes.

11.2 Adversarial Robustness of Learned Policies

Small perturbations of observations can degrade neural-network policies [62]. Policies with small, bounded, interpretable state vectors (for example, tabular or linear) have a smaller attack surface but are not invulnerable to sensor spoofing. Practical mitigations include plausibility checks on inputs, redundant sensing, limits on action changes, retraining on perturbed observations, and a safety supervisor that bounds the effect of any single malfunctioning action [30].

11.3 Privacy and Data Governance

Depending on the implementation, adaptive lighting and traffic-aware nodes may detect pedestrian or vehicle activity, which can raise privacy concerns. Data minimization should be the default, for example by using aggregate counts rather than identifiable data. Fleet learning can keep raw data on the node [56], [57], and with secure aggregation the server cannot inspect individual updates [63].

XII. DISCUSSION

RQ1 (RL families). Among the methods seen in the literature, tabular and linear methods are the most commonly implemented in harvesting-node power management, which suits low-end MCUs [33], [34], while deep value-based and actor–critic methods are more frequently trained off-device and deployed as compact networks [8], [38]. **RQ2 (applications and design).** The most direct application is energy-harvesting power management, with transferable formulations from traffic, building, and network control (Section 6.5). **RQ3 (on-device cost).** Although the memory, latency, and energy cost of the RL agent is deducted from the node’s budget [14], [18], most studies report control performance only, rather than quantifying the cost of the agent. **RQ4 (evaluation).** Simulation dominates, hardware validation is limited, and baselines and harvest traces are not standardized, which hampers comparison [47], [48]. **RQ5 (open issues).** These are discussed in Section XIII.

XIII. OPEN CHALLENGES

Sample efficiency. Deep and actor–critic methods are data-hungry and tuning-sensitive (Table 1), so nodes must rely on simulation pre-training, transfer, or shared policies [65]. **Nonstationarity.** Seasonal change, cloud regimes, and component ageing shift the dynamics, and policies must adapt without forgetting. **On-device learning cost.** Learning updates and memory access consume node energy (Eq. 7), and few studies quantify them; on-device learning frameworks for microcontrollers [58] are a starting point. **Safety.** Learned policies must never violate battery or service limits, which motivates constrained RL and supervisors [29], [30]. **Benchmarking.** There is no common suite of harvest traces, battery models, and baselines [47], and general-purpose TinyML benchmarks [64] do not cover energy-harvesting control.

XIV. RESEARCH ROADMAP

We define future work in terms of three horizons. **Near term (1–2 years).** Provide open benchmark suites containing realistic energy traces, standard energy and load models, and a common baseline suite, building on existing TinyML benchmarking efforts [64]; report on-device energy, memory, and latency of each policy; and assess over seasons and multiple random seeds. **Medium term (3–5 years).** Use on-device learning for drift [58] and deploy hybrid controllers consisting of a learned policy and a verified safety supervisor on real hardware; and investigate fleet-level learning using federated or offline RL under intermittent connectivity. **Long term (5+ years).** Achieve certifiable, explainable, and self-calibrating city-scale energy controllers, integrate lifecycle-aware goals such as battery ageing and embodied energy, and coordinate heterogeneous nodes with scalable multi-agent approaches.

Going forward, we suggest the following minimum reporting standard for future studies: (i) the data source used for harvest and load; (ii) the battery model and its parameters; (iii) at least one heuristic-based and one model-based baseline; (iv) the resource footprint on the device; and (v) results reported across multiple seeds and seasons.

XV. CONCLUSION

While the principles of RL are suitable for harvesting-powered smart-city edge nodes, its practicality depends on matching algorithm complexity to edge-node hardware, ensuring safety, and testing critically. This review contributed a taxonomy by edge deployability, a synthesis of applications, a layered reference architecture

instantiated for solar street lighting, and a prioritized roadmap. We hope that the proposed reporting standard and benchmark agenda will help move urban energy controllers from isolated simulations toward reproducible, deployable, and trustworthy systems.

REFERENCES

- [1]. A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.
- [2]. W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
- [3]. S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 3, pp. 443–461, 3rd Quart. 2011.
- [4]. A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Trans. Embedded Comput. Syst.*, vol. 6, no. 4, Art. 32, 2007.
- [5]. J. Hsu, S. Zahedi, A. Kansal, M. Srivastava, and V. Raghunathan, "Adaptive duty cycling for energy harvesting systems," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, 2006, pp. 180–185.
- [6]. C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Real-time scheduling for energy harvesting sensor nodes," *Real-Time Syst.*, vol. 37, no. 3, pp. 233–260, 2007.
- [7]. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
- [8]. V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [9]. K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.
- [10]. N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart. 2019.
- [11]. C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 3rd Quart. 2019.
- [12]. Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.
- [13]. X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart. 2020.
- [14]. R. David *et al.*, "TensorFlow Lite Micro: Embedded machine learning for TinyML systems," in *Proc. Mach. Learn. Syst. (MLSys)*, vol. 3, 2021, pp. 800–811.
- [15]. P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. Sebastopol, CA, USA: O'Reilly Media, 2019.
- [16]. M. J. Page *et al.*, "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," *BMJ*, vol. 372, Art. n71, 2021.
- [17]. B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele Univ., Keele, U.K., Tech. Rep. EBSE-2007-01, 2007.
- [18]. L. Lai, N. Suda, and V. Chandra, "CMSIS-NN: Efficient neural network kernels for Arm Cortex-M CPUs," arXiv:1801.06601, 2018.
- [19]. M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: Wiley, 1994.
- [20]. C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 279–292, 1992.
- [21]. J. R. Piorno, C. Bergonzini, D. Atienza, and T. S. Rosing, "Prediction and management in energy harvested wireless sensor nodes," in *Proc. Int. Conf. Wireless Commun., Veh. Technol., Inf. Theory Aerosp. Electron. Syst. Technol. (Wireless VITAE)*, 2009, pp. 6–10.
- [22]. O. Tremblay and L.-A. Dessaint, "Experimental validation of a battery dynamic model for EV applications," *World Electr. Veh. J.*, vol. 3, no. 2, pp. 289–298, 2009.
- [23]. H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.
- [24]. Z. Wang *et al.*, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 1995–2003.
- [25]. T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016.
- [26]. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv:1707.06347, 2017.
- [27]. T. Haarnoja *et al.*, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. ICML*, 2018, pp. 1861–1870.
- [28]. T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.
- [29]. J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. ICML*, 2017, pp. 22–31.
- [30]. M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2669–2678.
- [31]. A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 1179–1191.
- [32]. S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," arXiv:2005.01643, 2020.
- [33]. R. C. Hsu, C.-T. Liu, and H.-L. Wang, "A reinforcement learning-based ToD provisioning dynamic power management for sustainable operation of energy harvesting wireless sensor node," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 2, pp. 181–191, Jun. 2014.
- [34]. S. Shresthamali, M. Kondo, and H. Nakamura, "Adaptive power management in solar energy harvesting sensor node using reinforcement learning," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5s, Art. 181, 2017.
- [35]. S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. ICLR*, 2016.
- [36]. B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 2704–2713.
- [37]. V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

- [38]. F. A. Aoudia, M. Gautier, and O. Berder, "RLMan: An energy manager based on reinforcement learning for energy harvesting wireless sensor networks," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 2, pp. 408–417, Jun. 2018.
- [39]. T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1086–1095, Mar. 2020.
- [40]. J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, pp. 1437–1480, 2015.
- [41]. F. Fraternali, B. Balaji, D. Sengupta, D. Hong, and R. K. Gupta, "ACES: Automatic configuration of energy harvesting sensors with reinforcement learning," *ACM Trans. Sens. Netw.*, vol. 16, no. 4, Art. 36, 2020.
- [42]. H. Wei, G. Zheng, H. Yao, and Z. Li, "IntelliLight: A reinforcement learning approach for intelligent traffic light control," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 2496–2505.
- [43]. E. Mocanu *et al.*, "On-line building energy optimization using deep reinforcement learning," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3698–3708, Jul. 2019.
- [44]. Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1277–1290, Jun. 2019.
- [45]. H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. ACM Workshop Hot Topics Netw. (HotNets)*, 2016, pp. 50–56.
- [46]. J. R. Vázquez-Canteli and Z. Nagy, "Reinforcement learning for demand response: A review of algorithms and modeling techniques," *Appl. Energy*, vol. 235, pp. 1072–1089, 2019.
- [47]. G. Dulac-Arnold *et al.*, "Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis," *Mach. Learn.*, vol. 110, no. 9, pp. 2419–2468, 2021.
- [48]. P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 3207–3214.
- [49]. G. Brockman *et al.*, "OpenAI Gym," arXiv:1606.01540, 2016.
- [50]. T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. ICLR*, 2016.
- [51]. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.
- [52]. A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. ICML*, 1999, pp. 278–287.
- [53]. J. Tobin *et al.*, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2017, pp. 23–30.
- [54]. W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, 2020, pp. 737–744.
- [55]. A. A. Rusu *et al.*, "Policy distillation," in *Proc. ICLR*, 2016.
- [56]. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2017, pp. 1273–1282.
- [57]. W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart. 2020.
- [58]. H. Ren, D. Anicic, and T. A. Runkler, "TinyOL: TinyML with online-learning on microcontrollers," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2021, pp. 1–8.
- [59]. M. Horowitz, "Computing's energy problem (and what we can do about it)," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2014, pp. 10–14.
- [60]. E. F. Camacho and C. Bordons, *Model Predictive Control*, 2nd ed. London, U.K.: Springer, 2007.
- [61]. R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed Internet of Things," *Comput. Netw.*, vol. 57, no. 10, pp. 2266–2279, 2013.
- [62]. S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," arXiv:1702.02284, 2017.
- [63]. K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2017, pp. 1175–1191.
- [64]. C. Banbury *et al.*, "MLPerf Tiny benchmark," in *Proc. NeurIPS Datasets Benchmarks Track*, 2021.
- [65]. F. Fraternali *et al.*, "Scaling configuration of energy harvesting sensors with reinforcement learning," in *Proc. Int. Workshop Energy Harvesting Energy-Neutral Sens. Syst. (ENSys)*, Shenzhen, China, Nov. 2018 (arXiv:1811.11259).
- [66]. U. Y. Ogras *et al.*, "tinyMAN: Lightweight energy manager using reinforcement learning for energy harvesting wearable IoT devices," in *Proc. tinyML Res. Symp.*, 2022.
- [67]. M. Asif *et al.*, "Adaptive control of streetlights using deep learning for the optimization of energy consumption during late hours," *Energies*, vol. 15, no. 17, Art. 6337, 2022.