

INTERMITTENT-COMPUTING-AWARE DEEP LEARNING FOR BATTERYLESS IOT NODES: JOINT OPTIMIZATION OF INFERENCE SCHEDULING AND ENERGY HARVESTING DYNAMICS

Sharad Jagtap¹, Dr. G. Sudhagar², Sanjyot Patil³

¹Research Scholar, Dept. of ECE Engineering, BIHER, Chennai, India, Assistant Professor, Dept. of E & TC Engineering, SPPU, Pune, India, ORCID ID: 0009-0004-3059-5612, sharadjagtap0710@gmail.com

²Associate Professor, Dept. of ECE Engineering, Bharath Institute of Higher Education and Research, Chennai, India, ORCID ID: 0009-0002-5996-5694, sudhagar.ece@bharathuniv.ac.in

³Lecturer, Vidya Pratishthan's Polytechnic College, Pune, India, ORCID ID: 0009-0006-0189-5717, sanjyot.s.patil@gmail.com

ABSTRACT

A batteryless sensor node lives on whatever energy it can collect from its surroundings, and that supply rarely behaves. Running a neural network on such a node means coping with power failures in the middle of an inference, the cost of saving progress to non-volatile memory, and results that arrive too late to be of use. We describe a scheduler that treats these problems as a single decision: how deep an exit of a multi-exit network to run, and when to run it. It looks at the energy stored in the capacitor and a running estimate of the harvested power, keeps its spending in line with what the harvester can sustain, waits briefly when a deeper exit is within reach before the deadline, and declines events it cannot afford. Every inference it starts runs to completion without a power failure. In a 24-hour simulation of a solar-powered node, with mean harvested power between 57 and 908 microwatts, we compared it with three baselines. The classifier accuracies come from a public digit recognition data set, while the energy costs are assumed values. Counting correct and timely results per sensing event, the scheduler scored between 0.505 and 0.970, against 0.069 to 0.571 for a checkpointed full network, and it beat a greedy energy-aware policy by 0.9 to 4.9 points. Hardware measurements are still needed to confirm these results.

KEYWORDS: intermittent computing, batteryless IoT, energy harvesting, deep learning, multi-exit neural networks, TinyML, inference scheduling

INTRODUCTION

Sensor nodes that run on harvested light, radio waves or motion need no battery changes, which matters when thousands of them sit in fields, shops or factories [1]–[3]. Over the last few years TinyML has also made it realistic to run a neural network on a microcontroller, so a node can classify what it senses instead of streaming raw data [4], [5]. The two ideas fit together well on paper. In practice they clash: a neural network inference needs a fixed lump of energy delivered without interruption, while a harvester supplies a trickle that rises and falls with the weather.

Intermittent computing was developed for exactly this situation. Programs write their progress to non-volatile memory, or are split into small tasks, so that they can pick up again after the supply collapses [6]–[9], and a few systems apply the idea to neural network inference [10], [11]. The approach has a price. Saving and restoring state costs energy, a node that keeps browning out burns part of its harvest on rebooting, and an answer that appears after its deadline is often worthless. Early-exit networks [12] add another option, since a classifier can stop at a shallow exit when energy is short. What is usually missing is a link between that choice and the state of the harvester.

We close that link. The scheduler we propose picks the exit and the moment of execution together, so that the inference can finish on the energy already in the capacitor, or on what the harvester is likely to add before the deadline. It follows the energy-neutral idea of Kansal et al. [13]: over time, the node should not spend more per event than the harvester can replace. When the deadline leaves room, it waits for a deeper exit instead of settling at once for a shallow one.

Our contributions are threefold. First, we cast inference scheduling on a batteryless node as the problem of maximizing correct, on-time results subject to the capacitor and the deadline. Second, we give a simple online policy built from an energy budget, a bounded wait and an admission test; it needs one voltage reading and a few multiplications per event. Third, we compare the policy with three baselines in a day-long simulation at five harvesting levels, and we report effective accuracy, completion ratio, power failures and wasted energy.

Section II reviews related work. Sections III and IV give the system model and the proposed scheduler. Section V describes the simulation setup, Section VI discusses the results and their limits, and Section VII concludes.

Related Work

Intermittent Computing

Several systems keep a program moving across power failures. Mementos copies volatile state to non-volatile memory as the supply voltage falls [6]. Chain divides programs into tasks that exchange data through non-volatile channels [7]. Mayfly adds support for data that goes stale [8], and a survey by Lucia et al. lays out the open problems [9]. Closer to our topic, [10] and [11] run neural network inference on intermittently powered devices, the latter also dealing with timing constraints and on-device adaptation. These systems make an inference of fixed cost survive power failures. We ask instead which cost to pick, and when, so that the failures do not happen.

Resource-Constrained and Early-Exit Inference

A large body of TinyML work shrinks models and tunes runtimes for microcontrollers; [4] and [5] are recent overviews. BranchyNet showed that intermediate classifiers let easy inputs leave a network early [12]. Li et al. trained load-forecasting models on smart meters with only 192 KB of static memory by splitting the work between the meters, edge servers and the cloud [14]. In these works the power supply is generally assumed to be steady, so the energy available for one inference is not a quantity that changes at run time.

Energy-Harvesting IoT Nodes

On the hardware side, [1] and [2] review energy-aware node design and RF and multi-source harvesting. Šabović et al. built a feasibility model for LoRaWAN, BLE Mesh and 6TiSCH nodes around the worst-case bursts of consumption that the capacitor must ride out [3]. Giuliano et al. adapt the packet parameters of batteryless LoRa sensors to the stored energy [15], and la Rosa et al. demonstrate an energy-autonomous LoRaWAN node with a small photovoltaic cell and a 2 mF capacitor [16]. For ambient IoT devices that are often unavailable while they harvest, Wu et al. propose duty-cycled monitoring and device grouping to speed up inventory [17]. The notion of energy-neutral operation goes back to Kansal et al. [13]. Most of this work adapts communication or duty cycles. In the papers we surveyed, choosing the depth of a neural inference from the harvesting state received little attention, and that is the gap we address.

System Model And Problem Formulation

Node and Energy Model

Consider a node with a harvester, a power-management circuit and a capacitor of capacitance C . We write $E(t)$ for the stored energy, $ph(t)$ for the harvested power, η for the combined harvesting and conversion efficiency, and ps for the sleep power of the microcontroller. In a time step Δ the stored energy changes as

$$E(t+\Delta) = \min\{E_{\max}, E(t) + \eta ph(t)\Delta - ps\Delta - e(t)\} \quad (1)$$

Here $e(t)$ is the energy the computation draws in that step. The node runs only while the capacitor voltage stays above V_{off} . After a brownout it restarts once the voltage climbs back to V_{on} , which is higher than V_{off} . The corresponding energies are

$$E_x = \frac{1}{2}CV_x^2, x \in \{\text{off}, \text{on}, \text{max}\} \quad (2)$$

so the energy that can be spent on computation is $E(t) - E_{\text{off}}$.

Multi-Exit Inference and Sensing Events

The network has K exits. Exit k costs c_k and reaches accuracy a_k , and both grow with k . Sensing event n occurs at $t_n = nT$, and its result counts only if it arrives by $t_n + D$. A node that ignores its energy state runs the inference with checkpointing. Each checkpoint adds an overhead of a fraction β of the inference energy, and if the capacitor runs dry the node switches off, recharges to V_{on} , pays a restore cost and carries on, normally after the deadline has passed. A node that knows its energy can do better: whenever $E(t) - E_{\text{off}} \geq c_k + m$, where m is a safety margin, it can run exit k in one go and skip the checkpoint.

Problem Formulation

Let $z_{n,k}$ be 1 if event n is served at exit k and 0 otherwise. We want the expected number of correct results per event to be as large as possible,

$$\max (1/N) \sum_{n,k} a_k z_{n,k} \quad (3)$$

under the constraints

$$\sum_k z_{n,k} \leq 1, \quad t_n \leq t_n + D \quad (4)$$

$$E(t) \geq E_{\text{off}} \text{ for all } t \quad (5)$$

where t_n is the completion time of a served event. Constraint (5) rules out brownouts, which checkpointed execution accepts by design. The problem has to be solved online, because the future harvest is unknown and each choice changes what is left for later events.

Proposed Framework

The scheduler in Fig. 1 acts once per sensing event, and once per second while an inference is waiting. Its inputs are the capacitor voltage and an estimate of the harvested power, which can come from the charging slope of the capacitor or from a sense resistor at the harvester output.

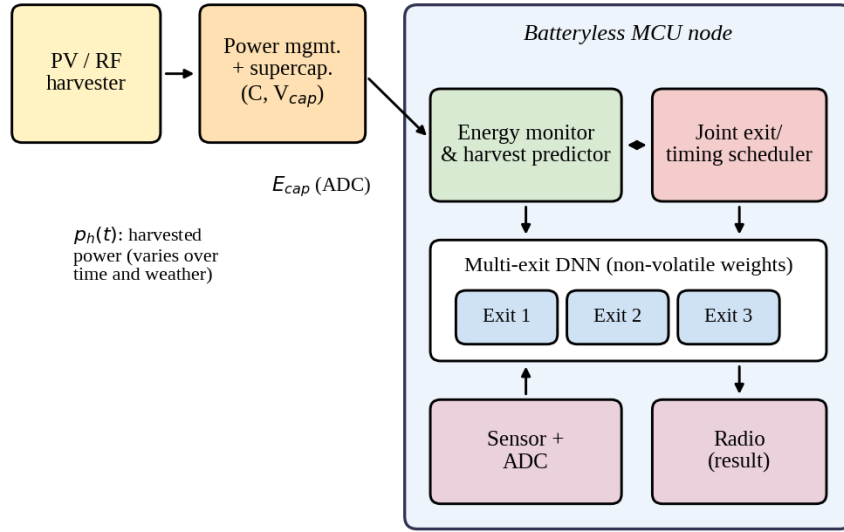


Fig 1. Block diagram of the proposed batteryless inference node.

Harvest Predictor

The scheduler tracks the harvested power with an exponentially weighted moving average,

$$q(t+1) = q(t) + \alpha(ph(t) - q(t)) \quad (6)$$

where α sets how quickly old samples fade. The average follows the daily rise and fall of light and smooths over short shadows, at the price of some lag. Any better predictor could take its place without touching the rest of the design.

Energy-Neutral Budget and Feasible Exits

To stay near energy-neutral operation [13], the scheduler first works out how much it may spend on the current event:

$$B(t) = \eta q(t)T - psT + (E(t) - Etgt)/H \quad (7)$$

The first two terms are the net energy expected from the harvester during one sensing period. The last term pulls the capacitor back toward a target level $Etgt$ over H events, so surplus charge gets used and a shortfall gets repaid. Energy that may still arrive before the deadline is counted conservatively:

$$G(t) = \rho(\eta q(t) - ps)D \quad (8)$$

where a ρ below one discounts the forecast. An exit is feasible if both $ck \leq B(t)$ and $ck + m \leq E(t) - Eoff + \max\{G(t), 0\}$ hold. Among the feasible exits the scheduler takes the deepest:

$$k^* = \max K(t) \quad (9)$$

Execution Rules

Four rules turn this choice into actions. (1) If no exit is feasible, the cheapest exit is still served when the stored energy exceeds its cost plus the margin by at least 1 mJ; otherwise the event is skipped and nothing is spent. (2) If $E(t) - Eoff \geq ck^* + m$, the inference starts at once and runs without a checkpoint. (3) If not, the event waits. Once per second the scheduler repeats the test and starts as soon as it passes. (4) When the deadline arrives, the scheduler runs the deepest exit the stored energy can then afford, or skips the event. Events that arrive while the node is off or busy are lost.

Overhead

Each decision costs at most K comparisons and a few multiply-accumulate operations for (6) to (8), which is negligible next to the inference. The only state the scheduler keeps is the scalar $q(t)$.

Experimental Setup

Workload and Exit Ladder

Our goal is to study scheduling, not a particular sensing task, so we use the 8×8 optical digit data set shipped with scikit-learn [18], [19] (1,797 images, ten classes) as a stand-in workload, with a stratified 60/40 train-test split. Three classifiers of growing capacity serve as exits: logistic regression on 4×4 average-pooled inputs, a 64-32-10 multilayer perceptron, and a 64-128-64-10 multilayer perceptron (Table I). They were trained separately, so they only approximate the exits of a jointly trained multi-exit network [12]. Accuracy was measured on the 719 held-out images.

The energy cost of each exit is not computed from these small models, whose cost would sit orders of magnitude below that of realistic TinyML workloads. We instead assign 0.25, 0.80 and 2.00 mJ. These values correspond to networks of roughly 0.4 to 3.3 million multiply-accumulate operations at an assumed 0.6 nJ per operation on a Cortex-M class microcontroller. They are assumptions and should be replaced by measurements.

Table 1. Exit Ladder: Measured Accuracy And Assumed Energy Cost

Exit	Classifier	Accuracy	c_k (mJ)
1	Logistic regression, 4×4 input	0.861	0.25
2	MLP 64-32-10	0.972	0.80
3	MLP 64-128-64-10	0.978	2.00

Accuracy is measured on 719 held-out images; energy costs are assumed values.

Harvesting Traces and Platform

We generated 24-hour solar traces at 1 s resolution. Power follows a daily envelope from 06:00 to 18:00 and drops to 5 percent of the peak at night, standing in for indoor or street lighting. A two-state Markov process switches between clear periods (mean 240 s) and shaded periods (mean 60 s, with 20 percent of the clear power), and log-normal noise of 10 percent is added. Peak powers of 0.2, 0.4, 0.8, 1.6 and 3.2 mW give mean harvested powers of 57, 113, 227, 454 and 908 μ W. For every level we simulated eight random realizations. Table II lists the platform and scheduler parameters.

Table 2. Platform And Scheduler Parameters

Parameter	Value
Capacitance C	2 mF
$V_{\text{off}} / V_{\text{on}} / V_{\text{max}}$	1.8 / 2.4 / 3.6 V
Efficiency η ; sleep power p_s	0.8; 3 μ W
Boot / restore energy	0.10 / 0.05 mJ
Checkpoint overhead β (baselines)	10% of c_k
Period T = deadline D	10 s
Safety margin m	0.2 mJ
Predictor α ; confidence ρ	0.01 per s; 0.8
Budget horizon H ; target E_{tgt}	10 events; $E_{\text{off}} + 0.4(E_{\text{max}} - E_{\text{off}})$
Simulation	24 h, 1 s step, 8 realizations

Baselines and Metrics

We compare four policies. CKPT-FULL always runs the deepest exit with checkpointing, and CKPT-SMALL always runs the smallest. GREEDY runs the deepest exit the stored energy can pay for at the moment of the event and skips the event otherwise, with no forecast, budget or waiting. PROPOSED is the scheduler of Section IV. All four share the same node model. Our main metric is effective accuracy, the average over all sensing events of the accuracy of the exit that delivered an on-time result, with lost, skipped and late events counted as zero. We also report the on-time completion ratio, power failures per day, and wasted energy as a percentage of the harvest. Wasted energy includes checkpoint overhead, restore and reboot energy, and the energy spent on results that missed their deadline. All numbers are averages over the eight realizations.

RESULTS AND DISCUSSION

Effective Accuracy

Fig. 2(a) and Table III show PROPOSED ahead at every harvesting level, from 0.505 at 57 μ W to 0.970 at 908 μ W. CKPT-FULL does worst (0.069 to 0.571). Its fixed 2.2 mJ per inference, inflated by checkpointing, keeps draining the capacitor; the node fails about 660 to 1,140 times a day, and most results are lost while it is busy or off, or arrive late. CKPT-SMALL avoids most failures once harvesting is moderate or high, but it cannot exceed the 0.861 accuracy of its only exit, and at low harvest it still fails 66 to 113 times a day. GREEDY uses stored energy for deeper exits and almost eliminates failures, yet it does not look ahead and never waits for a deeper exit. Against GREEDY, PROPOSED gains 0.019, 0.009, 0.049, 0.040 and 0.009 at the five levels. Against CKPT-SMALL it gains up to 0.109, at 908 μ W.

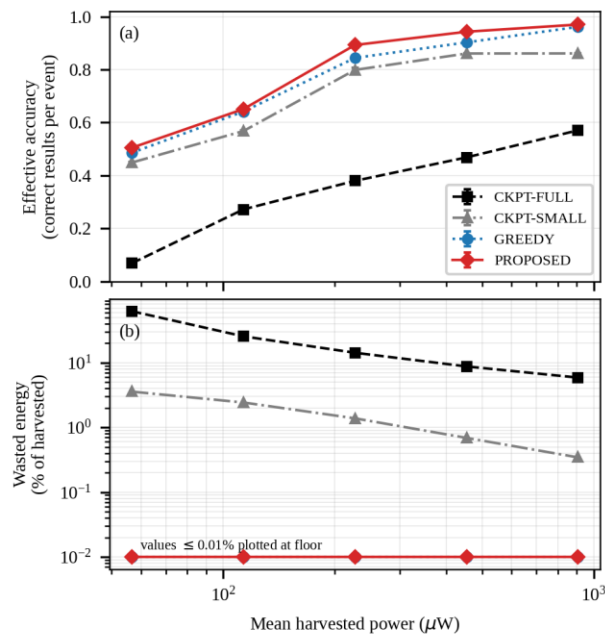


Fig 2. Effective accuracy (a) and wasted energy (b) versus mean harvested power, averaged over eight 24-hour realizations (error bars: standard deviation). In (b), GREEDY and PROPOSED coincide at the plotted floor.

Table 3. Results At Three Harvesting Levels (Mean Of Eight Realizations)

Policy	Effective accuracy	On-time ratio	Power failures per day	Wasted energy (%)
Mean harvested power: 57 μW				
CKPT-FULL	0.069	0.070	990	62.4
CKPT-SMALL	0.449	0.522	66.1	3.57
GREEDY	0.486	0.523	3.8	0.01
PROPOSED	0.505	0.543	0.0	0.00
Mean harvested power: 227 μW				
CKPT-FULL	0.380	0.389	676	14.2
CKPT-SMALL	0.799	0.928	38.4	1.38
GREEDY	0.844	0.915	0.0	0.00
PROPOSED	0.893	0.974	0.0	0.00
Mean harvested power: 908 μW				
CKPT-FULL	0.571	0.584	1,144	5.89
CKPT-SMALL	0.861	1.000	0.0	0.34
GREEDY	0.961	0.996	0.0	0.00
PROPOSED	0.970	1.000	0.0	0.00

Remaining levels are shown in Fig. 2.

Completion, Failures and Wasted Energy

The gain over GREEDY shows up in the completion ratio. At 227 μW , PROPOSED serves 97.4 percent of the events on time and GREEDY 91.5 percent: the budget spreads energy over time, and waiting turns some events that GREEDY would skip into served ones. PROPOSED pays for this with slightly shallower exits (mean exit index 1.87 against 1.96). At 908 μW both policies serve nearly every event, and PROPOSED is still ahead (0.970 against 0.961) with a lower mean exit index (2.44 against 2.54). This is consistent with the shape of the accuracy-energy curve in Table I: exit 3 adds only 0.6 accuracy points over exit 2 for 2.5 times the energy, so spreading energy evenly over events beats alternating between the deepest and the shallowest exit.

Fig. 2(b) shows wasted energy. CKPT-FULL wastes 62.4 percent of the harvest at 57 μW and 5.9 percent at 908 μW ; CKPT-SMALL wastes between 3.6 and 0.3 percent. GREEDY wastes under 0.01 percent and PROPOSED none, because neither runs checkpointed inference. PROPOSED had no power failure in any realization, whereas GREEDY averaged 3.8 failures a day at 57 μW . We suspect that GREEDY drains the capacitor to the safety margin and then runs into a shaded period, but we did not isolate this cause.

LIMITATIONS

These results come from a simulation with assumed parameters, so they should not be taken as hardware measurements. The per-exit energy costs, the checkpoint overhead and the boot and restore energies are all assumptions, and the ranking of the policies could shift once real values are used, especially the heavy penalty that CKPT-FULL pays for checkpointing. The exits are separately trained classifiers on a small digit data set, not a jointly trained multi-exit network on a real sensing task. The traces are synthetic, and the eight realizations share one daily envelope, so the reported spread (at most 0.007 standard deviation in effective accuracy) understates day-to-day and site-to-site variation. Inference is treated as instantaneous at 1 s granularity, sensing and radio energy are left out, and the scheduler is assumed to know the instantaneous harvested power. Finally, the scheduler parameters ($\alpha = 0.01$, $p = 0.8$, $H = 10$) were fixed in advance and not tuned for each harvesting level. A sensitivity study and a comparison with learned or optimal offline schedulers remain to be done.

CONCLUSION AND FUTURE WORK

We presented a scheduler for batteryless IoT nodes that chooses the exit of a multi-exit network and the time of execution together, using the stored energy and a forecast of the harvest. In a day-long simulation at mean harvested powers of 57 to 908 μW it gave the highest effective accuracy of the four policies at every level, with no power failures and no wasted energy, and it beat a greedy energy-aware policy by 0.9 to 4.9 points. Since the energy parameters are assumed, the next step is a hardware prototype on a microcontroller with non-volatile memory, with measured power profiles, real harvesting traces and a task-specific multi-exit network such as keyword spotting or shelf-stock monitoring. A sensitivity analysis and learned harvest predictors would follow.

ACKNOWLEDGMENT

This article is prepared as a research objective at Bharat Institute of Higher Education

REFERENCES

1. Kanoun et al., “Energy-aware system design for autonomous wireless sensor nodes: A comprehensive review,” *Sensors*, 2021.
2. G. Moloudian et al., “RF energy harvesting techniques for battery-less wireless sensing, Industry 4.0, and Internet of Things: A review,” *IEEE Sensors J.*, 2024.
3. A. Šabović et al., “Energy harvesting for wireless IoT use cases: A generic feasibility model and tradeoff study,” *IEEE Internet Things J.*, 2023.
4. J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, and S. Han, “Tiny machine learning: Progress and futures,” *IEEE Circuits Syst. Mag.*, 2024.
5. S. Somvanshi et al., “From tiny machine learning to tiny deep learning: A survey,” *ACM Comput. Surv.*, 2025.
6. B. Ransford, J. Sorber, and K. Fu, “Mementos: System support for long-running computation on RFID-scale devices,” in *Proc. ASPLOS*, 2011.
7. A. Colin and B. Lucia, “Chain: Tasks and channels for reliable intermittent programs,” in *Proc. ACM OOPSLA*, 2016.
8. J. Hester, K. Storer, and J. Sorber, “Timely execution on intermittently powered batteryless sensors,” in *Proc. ACM SenSys*, 2017.
9. B. Lucia, V. Balaji, A. Colin, K. Maeng, and E. Ruppel, “Intermittent computing: Challenges and opportunities,” in *Proc. SNAPL*, 2017.
10. G. Gobieski, B. Lucia, and N. Beckmann, “Intelligence beyond the edge: Inference on intermittent embedded systems,” in *Proc. ASPLOS*, 2019.
11. B. Islam and S. Nirjon, “Zygarde: Time-sensitive on-device deep inference and adaptation on intermittently-powered systems,” *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2020.
12. S. Teerapittayanon, B. McDanel, and H. T. Kung, “BranchyNet: Fast inference via early exiting from deep neural networks,” in *Proc. Int. Conf. Pattern Recognit. (ICPR)*, 2016, pp. 2464–2469.
13. A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, “Power management in energy harvesting sensor networks,” *ACM Trans. Embed. Comput. Syst.*, vol. 6, no. 4, 2007.
14. Y.-H. Li et al., “Introducing edge intelligence to smart meters via federated split learning,” *Nature Commun.*, 2024.
15. F. Giuliano et al., “Adaptive algorithms for batteryless LoRa-based sensors,” *Sensors*, 2023.
16. R. la Rosa et al., “Towards mass-scale IoT with energy-autonomous LoRaWAN sensor nodes,” *Sensors*, 2024.
17. Z. Wu et al., “Fast inventory for 3GPP ambient IoT considering device unavailability due to energy harvesting,” *IEEE Commun. Standards Mag.*, 2025.
18. F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
19. E. Alpaydin and C. Kaynak, “Optical recognition of handwritten digits,” *UCI Machine Learning Repository*, 1998.
20. Y. Wang et al., “Deep learning on energy harvesting IoT devices: Survey and future challenges,” *IEEE Access*, vol. 10, pp. 124999–125014, 2022. doi: 10.1109/ACCESS.2022.3225092.

21. 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.
22. L. Lai, N. Suda, and V. Chandra, "CMSIS-NN: Efficient neural network kernels for Arm Cortex-M CPUs," *arXiv:1801.06601*, 2018.
23. J. Lin et al., "MCUNet: Tiny deep learning on IoT devices," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.
24. H. Cai et al., "TinyTL: Reduce memory, not parameters for efficient on-device learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.
25. C. Banbury et al., "MicroNets: Neural network architectures for deploying tinyML applications on commodity microcontrollers," in *Proc. Machine Learning and Systems (MLSys)*, 2021.
26. H. Cai et al., "TinyNAS: Neural architecture search on tiny devices," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.
27. B. Ransford, J. Sorber, and K. Fu, "Mementos: System support for long-running computation on RFID-scale devices," in *Proc. ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2011.
28. M. Uko et al., "Perpetual edge intelligence: Adaptive hybrid energy harvesting and reinforcement-learning-based TinyML for autonomous IoT sensors," *Discover Electronics*, vol. 3, Art. no. 33, 2026.