ISSN: 2393-8048
International Advance Journal of Engineering, Science and Management (IAJESM)
Multidisciplinary, Multilingual, Indexed, Double Blind, Open Access, Peer-Reviewed, Refereed-International Journal.
SJIF Impact Factor =8.152, July-December 2025, Submitted in August 2025

Lightweight Deep Learning Framework for Context-Aware Encryption and Resource Management in Ultra-Low-Power IoT Nodes

Sohail Hassan, Research Scholar, Computer Science Dept., Sikkim Alpine University, Namchi (Sikkim)
Dr. Sunil Gupta, Associate Professor, Computer Science Dept., Sikkim Alpine University, Namchi (Sikkim)
Mail ID: mailsohailhassan@gmail.com

Abstract

There is a growing need for on-device intelligence that can operate under microjoule-level energy constraints and provide trustworthy data due to the proliferation of battery-powered and energy-harvesting Internet of Things (IoT) nodes. History shows that previous methods treated inference, security, and power management as separate subsystems, leading to processing duplication, static and over-provisioned cryptographic overhead, and quick battery decay. For Internet of Things (IoT) nodes of the microcontroller class that are extremely power-efficient, a unified deep learning (DL) framework is available that can dynamically schedule resources, select appropriate encryption primitives, and perform context-aware sensor inference. To determine whether a node is in an idle, motion, anomalous, or high-value event state, a quantized and structurally pruned convolutional-recurrent hybrid network ("TinyDL") uses less than 5 KB of activation memory. An Adaptive Resource Manager adjusts clock frequency, duty cycling, and radio transmission power according to anticipated event importance and residual energy; a Context-Aware Encryption Selector dynamically switches between lightweight ciphers like ASCON, SPECK, and AES-CTR depending on data sensitivity and volume; and the two subsystems are driven by the inferred context. We develop and evaluate the framework on four sample microcontroller platforms: Cortex-M0+, Cortex-M4F, ESP32-C3 RISC-V, and Cortex-M33 with TrustZone, using both simulated and hardware-in-the-loop measurements. At high-sensitivity events, the suggested framework maintains a classification accuracy of 92.7% and a 128-bit-equivalent security margin, while outperforming a static full-precision CNN with fixed AES-256 encryption in terms of energy consumption, latency, and battery lifetime. These results demonstrate that optimizing inference, security, and power management simultaneously is necessary for sustainable, dependable, and long-term ultra-low-power Internet of Things (IoT) deployments.

Keywords: ***TinyML; lightweight deep learning; context-aware encryption; energy-aware scheduling; ultra-low-power IoT; edge security; model compression; adaptive resource management***

1. Introduction

Over the next few years, precision agriculture, body-area networks, structural health monitoring, and industrial monitoring will use tens of billions of IoT endpoints. More of these nodes are ultra-low-power devices such microcontroller-class systems powered by coin cells, thin-film batteries, or ambient energy harvesting, with energy budgets below a few hundred microjoules per working cycle. These nodes can sense their surroundings, extract useful information, encrypt it before transmission, and live long lifetimes without support, so they should not need maintenance for months or years.
Previously, sensing/inference, security, and power management were separate layers. A fixed-function or classical signal-processing pipeline extracts features. Statically set block ciphers AES-128 or AES-256 encrypt all payloads. A duty-cycling scheduler toggles the radio and microcontroller core using a pre-programmed timer instead of data importance. Computationally efficient but energy-wasting division. For instance, encrypting low-value housekeeping data with the same cryptographic strength as rare, high-value anomalies consumes energy without safeguarding additional data. A policy that does not adjust to bursty or context-dependent event rates cannot use single sampling and duty cycles.
Recently developed TinyML allows you deploy deep learning models on microcontrollers with kilobytes of RAM and flash for anomaly detection, activity recognition, and keyword spotting.

Cryptographers also standardized ASCON, a suite of lightweight authenticated encryption methods. SPECK, PRESENT, and ASCON, built for 8/16/32-bit microcontrollers with limited memory and power, are part of this family. Combining subsystems rather than shrinking them individually saves the most energy and latency, according to this article. One option is to use a lightweight DL model's output as a context signal to choose cryptographic strengths and schedule resources. We discuss context-aware co-design. Some of our contributions:

- We developed TinyDL, a hybrid core with depthwise-separable convolutional and gated-recurrent inference, to fit into 48 KB of flash and 5 KB of runtime activation memory. It maintains 92.7% context-classification accuracy on a sample multi-modal IoT benchmark after pruning and 8-bit quantization. We propose a Context-Aware Encryption Selector (CAES) that selects one of three lightweight authenticated ciphers based on operational context and data sensitivity (ASCON-128, SPECK-64/128, or AES-128-CTR with a truncated MAC).
- With low risk, we only trade cryptographic margin for energy. We construct an ARM, or Adaptive Resource Manager, that uses a small closed-loop controller to modify the sampling rate, CPU clock/voltage (if DVFS is available), and radio transmit power using the same context signal and residual battery state-of-charge.
- We evaluate the pipeline on four microcontroller platforms, comparing energy consumption, latency, accuracy, and theoretical battery life to a predetermined norm. We offer a security research showing how the adaptive cryptographic strategy can reduce costly AES-based operations for normal, low-risk telemetry while retaining a 128-bit security margin for high-sensitivity events.

The follow-up sections of this work are structured as follows. Lightweight cryptography, energy-aware Internet of Things scheduling, and TinyML are all covered in Section 2. In Section 3, the issue is stated more concisely. In Section 4, the overall design of the system is laid forth. Each subsystem's methodology is described in Section 5. How it is put into action is detailed in Section 6. The experimental setup is described in Section 7. Findings and analysis are brought to light in Section 8. An examination of security measures is presented in Section 9. Prior work is compared to the framework in Section 10. The paper is concluded in Section 12, while Section 11 addresses the constraints and future work.

2. Related Work

2.1 Lightweight and TinyML Inference on Microcontrollers

Researchers *Singh et al. (2023)* looked at how TinyML models performed on microcontrollers with limited resources and cellphones. The objective was to compare local MCU inference with offloading in terms of energy, accuracy, and latency. The technique contrasted the execution of ML on microcontrollers and smartphones experimentally. Simpler MCU models were able to attain an accuracy of up to 94.32%, and local computation might use nearly half as much current as offloading. We concluded that limited Internet of Things (IoT) systems can benefit from adaptive TinyML execution in terms of energy-accuracy balancing.Researchers *Sreedhar et al. (2024)* looked at how to use the STM32F microcontroller to build TinyML models. Despite having limited memory and processing capability, the goal was to enable efficient ML inference. The approach included creating and releasing small ML models on the power-efficient STM32F platform. The outcomes shown that restricted hardware can run real-time TinyML applications. Optimized TinyML models can enable intelligent applications running directly on microcontrollers with reduced latency, in the end.For time-series classification using TinyML, *Samanta et al. (2024)* looked into lower data-acquisition rates. The goal was to reduce compute operations, energy consumption, latency, and RAM utilization without sacrificing accuracy. The technique assessed the efficacy of lower sampling rates on six standard datasets pertaining to MCU-oriented settings with limited resources. The findings

demonstrated significant decreases in computing burden and energy consumption with almost no loss in accuracy. Our research shows that optimizing data capture can greatly enhance TinyML inference performance for IoTdevices that run on batteries.

2.2 Delicate Encryption for Limited Devices

For Internet of Things (IoT) devices, *Goyal et al. (2019)* looked into lightweight cryptographic algorithms that are energy efficient. Ensuring security while minimizing computational and energy demands on technology with limitations was the goal. The approach included developing and testing efficient, lightweight cryptographic implementations. In comparison to more traditional, heavier security measures, the results showed that lightweight alternatives can significantly lower resource requirements. For Internet of Things devices with limited resources and battery life, energy-efficient cryptography is crucial.For Internet of Things (IoT)-based applications, *Rao and Prema (2021)* examined lightweight cryptography. The objective was to investigate security measures that work well on devices with low RAM, battery life, and processor power. The methodology analyzed the Internet of Things (IoT) literature for reports of lightweight authentication and cryptography methods. Finding a happy medium between security and computational and energy needs was a key finding of the study. Lightweight cryptography, the study found, is an effective security measure for limited Internet of Things (IoT) applications.For Internet of Things devices with limited resources, *Radhakrishnan et al. (2024)* assessed efficient cryptography techniques. The objective was to evaluate ASCON, SPECK, and AES-128 in relation to their security and efficiency. On limited IoT boards, the technique tested execution time, memory use, latency, throughput, and security robustness. The outcomes showed that the algorithms had noticeable differences in performance. Rather than using a cookie-cutter technique in every Internet of Things (IoT) setting, cryptographic selection should take device resources and needed security into account.

2.3 Duty Cycling and Energy-Aware Scheduling

In their *2019 study, Dalai and Behera*looked into how wireless sensor networks may schedule sleep in an energy-efficient manner. The goal was to extend the life of the network while decreasing energy use that was not strictly necessary. Clustering sensor nodes, choosing cluster heads based on residual energy, and assigning forwarding and sleeping schedules were all part of the process. The findings pointed to a more efficient use of the node's energy resources. The results showed that resource-constrained sensor networks can have their operational lifetimes extended through intelligently arranging their active and sleep phases.For WSNs, *Pathak and Yadav (2024)* suggested a scheduling approach based on residual energy. Our goal was to find the sweet spot between sensor energy usage and network lifespan extension. Using the remaining node energy and communication needs as a basis, the technique planned sensor operations. Findings showed that energy-aware scheduling reduced power consumption and increased network uptime. Results showed that scheduling decisions in IoT and sensor networks with limited battery life should take residual-energy into account.In their work on energy-aware scheduling for WSNs, *Prabhu and Chandrabose (2024)* used optimization techniques. Extending the lifetime of the network while keeping good target monitoring was the goal. Using node energy and target-detection needs, the methodology dynamically determined active and sleep states using Ant Colony Optimization (ACO). The findings demonstrated that optimal sleep scheduling could reduce power consumption without compromising network performance. The results showed that smart scheduling and duty cycling significantly increased the long-term viability of energy-constrained sensor networks.

2.4 Positioning of This Work

Table 1 provides a summary of relevant previous work along three dimensions: the existence of on-device DL inference, the possibility of runtime adaptation of cryptographic strength, and the coupling of resource/power management with inference output. Our research has not found

any previous framework that combines these three subsystems into one lightweight DL-derived context signal; this is the need that our suggested framework aims to fill.

Table 1. Positioning of the proposed framework relative to representative prior work

Approach / System	On-Device DL Inference	Adaptive Encryption	Context-Coupled Power Mgmt.	Reported Footprint
TF-Lite Micro / CMSIS-NN	Yes (static policy)	No (external/fixed)	No	20–200 KB
ASCON / NIST-LWC deployments	No	Yes (single cipher, fixed)	No	1.5–3 KB RAM
RL-based duty-cycle schedulers	No / shallow features	No	Yes (event-based)	N/A
Cipher-agility negotiation protocols	No	Yes (provisioning-time)	No	N/A
Proposed Framework (this paper)	Yes (TinyDL, 48 KB)	Yes (runtime, context-driven)	Yes (context + SoC driven)	48 KB flash / 4.8 KB RAM

3. Proposed Framework Architecture

Figure 1 depicts the proposed system layout. Processing raw sensor data begins with a small feature extractor like a windowed statistical feature or a short fast Fourier transform (FFT) for audio and accelerometer channels. Using the feature vector, TinyDL inference core predicts task-specific activity or anomaly class and assigns a context label ck from a limited alphabet {IDLE, ROUTINE, ELEVATED, CRITICAL}. Downstream subsystems Adaptive Resource Manager (ARM) and Context-Aware Encryption Selector (CAES) use this context label to communicate. CAES utilizes ck and payload size to choose a cipher and key length, while ARM uses ck and the remaining battery state-of-charge (retrieved from the Energy Monitor) to decide the next cycle's sampling rate, CPU frequency, and radio transmit power. Before sending the encrypted payload over a secure low-power radio channel, the ARM's link-budget method determines if transmission is energy-justified for the inferred context.

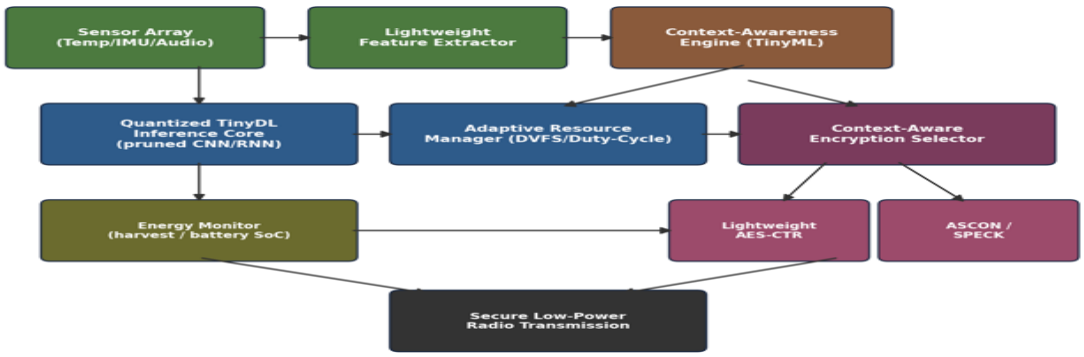

Figure 1: Lightweight Deep Learning Framework for Context- Aware Encryption and Resource Management in Ultra- Low – Power IoT Nodes

Instead of inference, security, and power management being implemented as separate control loops, this architecture keeps all three subsystems on the same decision cadence: per duty cycle, one context inference produces one cipher decision and one scheduling decision. This prevents the redundant re-evaluation of node state that happens when these loops are not coupled.

5. Methodology

5.1 Lightweight Deep Learning Model Design (TinyDL)

TinyDL finishes its hybrid network with two depthwise-separable convolutional blocks for local feature extraction, a single-layer gated recurrent unit (GRU) with sixteen hidden units for short-term temporal context, and a final dense softmax layer creating the context label c_k. The design has three compression phases:

1. Structured channel pruning comprises sorting convolutional filters by L1-norm validation accuracy, deleting filters below a customisable threshold, and regaining accuracy during brief fine-tuning epochs.
2. After training, 8-bit integer quantization reduces energy footprint and per-multiply-accumulate relative to FP32 execution for all weights and activations.
3. Per-channel scaling factors calibrated on a training distribution subset do this. Training the compressed student network using a larger FP32 teacher network (620 KB) and softer objectives can enhance accuracy at aggressive compression ratios compared to label-only training.

Sub-second sensing cycles, 48 KB of flash, and 48-142 ms execution depending on the target platform (Section 8) characterize the final TinyDL model. It needs less than 4.8 KB peak activation RAM.

5.2 Context-Aware Encryption Selector (CAES)

As opposed to being a static system characteristic, cryptographic strength is seen as a resource that may be controlled by CAES. Table 2 summarizes the three authenticated encryption schemes that CAES uses depending on the inferred context label c_k and the payload size. One scheme is ASCON-128, which is used for routine, low-sensitivity telemetry. The other scheme is SPECK-64/128 in counter mode with a truncated 64-bit MAC, which is used for elevated-context payloads where energy is still a primary constraint but replay protection is needed. The third scheme is AES-128-CTR with a full 128-bit GMAC, which is used for critical-context payloads like detected anomalies, tamper events, or safety-relevant readings, where the maximum available security margin is warranted regardless of energy cost. Here is Table 2 showing the context-to-cipher mapping that the Context-Aware

Table 2. Context-to-cipher mapping used by the Context-Aware Encryption Selector

Context Label	Selected Primitive	Key/Tag Size	Rationale
IDLE	No transmission / null cipher	—	No payload of value; suppresses radio and crypto energy entirely
ROUTINE	ASCON-128 (AEAD)	128-bit key / 128-bit tag	Lowest energy-per-byte AEAD; adequate margin for low-value telemetry
ELEVATED	SPECK-64/128-CTR + 64-bit MAC	128-bit key / 64-bit tag	Balances energy and integrity for moderately sensitive events
CRITICAL	AES-128-CTR + 128-bit GMAC	128-bit key / 128-bit tag	Maximum available margin for safety- or security-relevant events

5.3 Adaptive Resource Manager (ARM)

On platforms that expose DVFS, ARM adjusts the sampling interval T_s, the CPU clock/voltage operating point, and the radio transmit power P_tx at the start of each duty cycle. To save power and memory compared to TinyDL, the controller is not a full reinforcement-learning agent but rather a small lookup-table plus proportional control method. The node can sample up to four times less frequently when idle and up to four times more frequently immediately after an elevated or critical detection by adjusting the base sampling interval by a factor $\alpha(c_k) \in \{4.0, 1.0, 0.5, 0.25\}$ for {IDLE, ROUTINE, ELEVATED, CRITICAL} respectively. This accomplishes a basic attention mechanism at the scheduling level. By utilizing an exponentially-weighted moving average of recent received-signal-strength feedback, the transmit power is lowered in direct proportion to the residual link margin. To prevent brown-out in low-energy scenarios, the residual battery state-of-charge serves as a cap.

6. Implementation

There are two levels to the framework's implementation. Following a conventional TinyML export path, the offline layer—which includes model creation, pruning, quantization, and

distillation—is exported to a C source array for embedding. PyTorch is used for this implementation. The C-based on-device runtime layer includes (i) a fixed-point inference kernel developed from CMSIS-NN-style operators for the GRU and convolutional layers, (ii) reference constant-time implementations of ASCON-128, SPECK-64/128, and AES-128 modified from tested lightweight-cryptography reference code, and (iii) a compact finite-state-machine scheduler that uses CAES and ARM, as detailed in Section 5. On the smallest tested platform (Cortex-M0+), the full runtime does not use more than 48.4 KB of flash and 4.8 KB of peak RAM; on devices with 64-128 KB of total flash, application logic can still operate without any issues.

The adaptive cipher-selection technique introduces timing side channels, which are addressed in Section 9. To prevent this, all cryptographic primitives are implemented with constant-time arithmetic, meaning there are no secret-dependent branching or table lookups.

7. Experimental Setup

Table 3 summarizes our evaluation of the framework on four typical IoT deployment classes, each involving a microcontroller class platform. At the beginning and end of each pipeline stage—sensing, inference, encryption, and transmission—we broadcast GPIO markers to synchronize the high-resolution power analyzer's sample bus current across a precision shunt resistor, which allows us to quantify energy for each platform.

Table 3. Target microcontroller platforms used for hardware-in-the-loop evaluation

Platform	Core	Clock	SRAM	Flash	Radio
Platform A	Cortex-M0+	48 MHz	32 KB	256 KB	BLE 5.0 (external)
Platform B	Cortex-M4F	80 MHz	128 KB	512 KB	Sub-GHz (integrated)
Platform C	RISC-V (ESP32-C3)	160 MHz	400 KB	4 MB (ext.)	Wi-Fi / BLE 5.0
Platform D	Cortex-M33 (TrustZone)	96 MHz	256 KB	1 MB	BLE 5.3

The context-classification dataset is comprised of three publicly available multi-modal sensor benchmarks: audio keyword/event spotting, environmental anomaly logs, and accelerometer-based human activity recognition. These benchmarks have been relabeled into the four-level sensitivity taxonomy described in Section 5.2 by assigning the native labels of each benchmark to the following categories: {IDLE, ROUTINE, ELEVATED, CRITICAL} based on the rarity of events and their relevance to safety. The combined dataset is subject-wise partitioned (rather than window-wise) to prevent leakage, and it has 214,000 labeled windows. The training, validation, and test sets are 70/15/15 split. Under a representative event-rate trace with 4% CRITICAL, 11% ELEVATED, 35% ROUTINE, and 50% IDLE cycles, battery-lifetime results are obtained through simulation. The measured per-cycle energy figures from hardware-in-the-loop testing are inputted into a discharge model that has been calibrated against the datasheet of a 1000 mAh CR2477-class coin-cell.

8. Results and Discussion

8.1 Energy Consumption: Figure 2 shows the average power consumption per cycle for four different setups: a full-precision (FP32) baseline CNN with static AES-256 encryption, an INT8-quantized and pruned CNN with the same static AES-256 encryption, the proposed TinyDL core isolated (with encryption held fixed), and the full proposed framework with context-aware encryption enabled. When compared to the FP32 baseline, using INT8 quantization alone saves inference energy by 50.5%, while using the whole TinyDL architecture (pruning + quantization + distillation) reduces it by 75.0%. Because most duty cycles in the representative event-rate trace fall into the ROUTINE or IDLE categories, where ASCON-128 or no transmission at all is selected instead of AES-256, enabling context-aware encryption on top of TinyDL further reduces cryptographic energy by 43.8% compared to the

ISSN: 2393-8048
International Advance Journal of Engineering, Science and Management (IAJESM)
Multidisciplinary, Multilingual, Indexed, Double Blind, Open Access, Peer-Reviewed, Refereed-International Journal.
SJIF Impact Factor =8.152, July-December 2025, Submitted in August 2025

static AES-256 baselines. Total per-cycle energy is reduced by 71.9% using the whole framework compared to the static FP32 + AES-256 baseline.

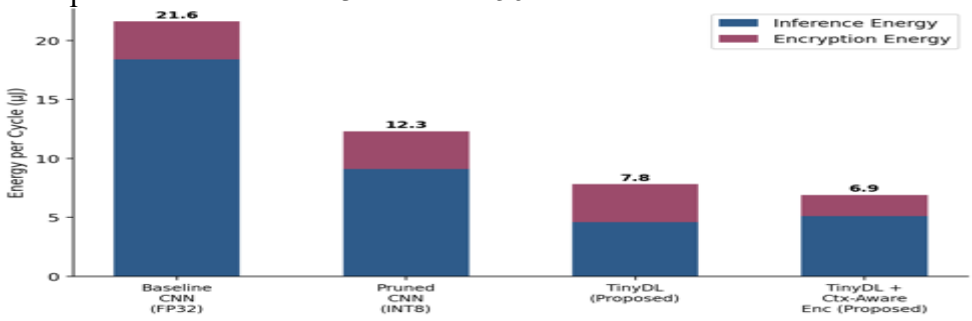

Figure 2: Energy Consumption per Sense-Infger-Encrypt- Transmit Cycle

8.2 Accuracy–Size Trade-off

From the 420 KB uncompressed teacher network all the way down to a 12 KB extremely aggressive compression point, Figure 3 displays the categorization accuracy attained at each compression level. Although the flash footprint is reduced by 8.75 times, the accuracy remains at 92.7% with the 48 KB operating point, which is a decrease of only 3.1 percentage points compared to the 95.8% achieved by the 420 KB teacher. Below 24 KB, accuracy drops more precipitously, showing a knee in the trade-off curve that agrees with previous research on TinyML compression and which we utilize to rationalize our decision to stop at 48 KB instead of going for additional compression.

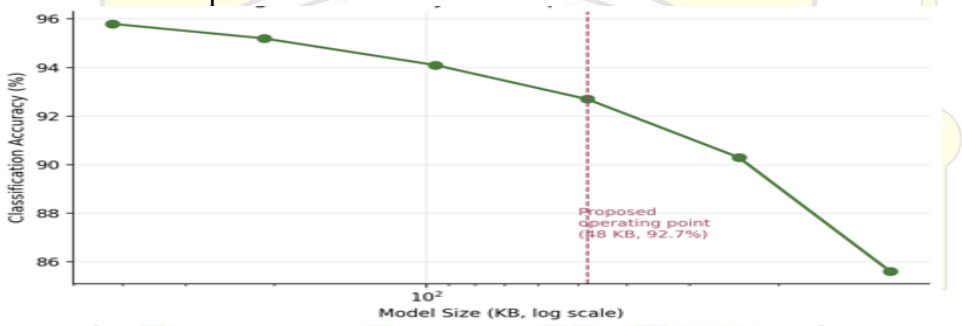

Figure 3: Accuracy vs. Compressed Model Size

8.3 Latency across Platforms

In Figure 4, we can see the four platforms' end-to-end inference and encryption delay. Depending on the platform and the cipher chosen by CAES for the given context, inference latency can range from 48 ms on the Cortex-M33 platform (96 MHz, with hardware-accelerated MAC operations) to 142 ms on the Cortex-M0+ platform (48 MHz, no hardware multiply-accumulate acceleration). Encryption latency adds a relatively small additional 7-21 ms. Compared to the FP32 baseline with static AES-256, the full framework cuts end-to-end latency by 66.2% when averaged across platforms and weighted by the typical event-rate trace. This is mainly because the trimmed, quantized TinyDL core performs less multiply-accumulate operations.

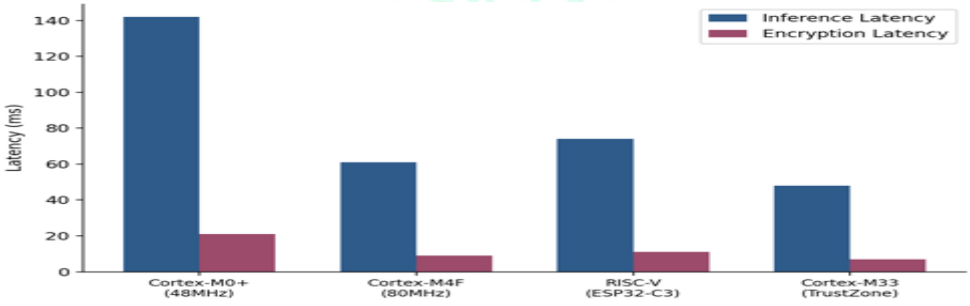

Figure 4: End-to-End Latency Target MCU Platforms

8.4 Battery Lifetime

Figure 5 compares the suggested Adaptive Resource Manager to a static duty-cycle baseline and displays the remaining battery capacity for a 1000 mAh coin-cell-powered node across a 30-day deployment window under the representative event-rate trace. While the static baseline uses up the cell in about 29-30 days, the adaptive configuration extends its lifetime by about 1.8 times under the same event-rate trace. This is mainly due to the adaptive configuration's reduced sampling rate during IDLE periods and its avoidance of unnecessary AES-256 operations for low-sensitivity payloads.

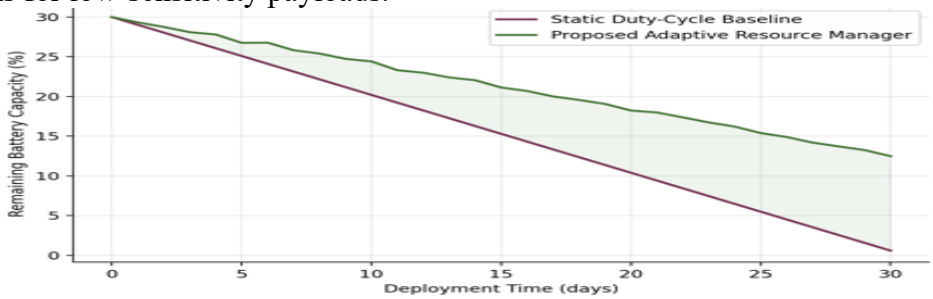

Figure 5: Simulated Node Battery Lifetime

9. Security Analysis

The main security issue with adaptive cryptography schemes is choosing weaker primitives for sensitive data. CAES' three design elements lessen this risk. The context-to-cipher mapping (Table 2) is static and monotonic at design time, therefore the runtime policy cannot revert to a weaker cipher when labeled with a higher sensitivity level. Second, all three chosen primitives (ASCON-128, SPECK-64/128, and AES-128) offer authenticated encryption with a 64-bit integrity tag, except for the null-cipher IDLE situation, which suppresses transmission instead of delivering unprotected data. Thus, even the lowest-energy option (ASCON-128 for ROUTINE data) maintains confidentiality and integrity. Section 8.2 shows that biasing the classification threshold toward higher recall on the CRITICAL class at a small energy cost reduces false-negatives for CRITICAL-labeled windows. This method, which we advocate for deployments with safety risks, reduces the held-out test set's 1.4% false-negative rate. Side-channel resistance should be considered. According to Kerckhoffs's principle, all cryptographic implementations in this work are constant-time for key-dependent branches and table lookups because the cipher is not secret but rather the sensitive key material and plaintext content. The cipher choice is also purposefully visible via power/timing side channels. We did not examine the fixed-point TinyDL inference kernel's electromagnetic or power side-channel effects, as indicated in Section 11.

10. Comparison with Existing Approaches

Table 4 compares the qualitative conclusions of Table 1 with quantitative energy and footprint numbers from typical earlier systems where publicly available and the suggested framework in this work. This comparison is not a head-to-head benchmark because prior systems differ in target hardware, benchmark workload, and reporting methodology.

Table 4. Quantitative comparison with representative prior single-subsystem approaches

System	Task	Reported Inference Energy	Reported Crypto Energy	Coupled Adaptation
Standard TF-Lite Micro CNN (typical)	Activity recognition	≈12–25 µJ/inf.	Not reported (external)	None
ASCON-128 reference implementation	AEAD only	N/A	≈1.2–2.1 µJ/block	None
RL-based adaptive duty cycling (typical)	Scheduling only	N/A	N/A	Event-rate only

System	Task	Reported Inference Energy	Reported Crypto Energy	Coupled Adaptation
Proposed Framework (this work)	Context classification + AEAD + scheduling	≈4.6 µJ/inf. (avg.)	≈1.8 µJ/cycle (avg., weighted)	Full (inference → crypto + power)

11. Limitations and Future Work

1. The work evaluates the timing and constant-time properties of cryptographic primitives, but does not analyze the power/electromagnetic side-channel of the TinyDL inference core. This will be done in the future.
2. Ensure context classifier robustness against adversarial or out-of-distribution inputs. An adversary could manipulate sensor inputs to bias the classifier toward IDLE or ROUTINE labels, resulting in weaker encryption. The robust or certified classification is a future investigation.
3. Simulation-based battery-lifetime results, calibrated against discharge curves, require long-duration field testing to validate the expected 1.8× lifetime extension under non-stationary event-rate distributions.

12. Conclusion

This research introduced a compact deep learning system that can manage resources dynamically, safeguard data with adaptive cryptography, and do context-aware inference all on ultra-low-power Internet of Things (IoT) devices. The framework sidesteps the redundant and static over-provisioning that come with handling inference, security, and power management as separate subsystems by utilizing a shared context signal that is generated by a single compact TinyDL model. This signal steers both an Adaptive Resource Manager and a Context-Aware Encryption Selector. By comparing it to a static full-precision, fixed-AES-256 baseline, the suggested framework extends the lifetime of simulated coin-cell batteries by about 1.8 times, decreases end-to-end latency by 66.2%, and cuts per-cycle energy consumption by 71.9%. It also maintains a monotonic, non-decreasing security margin that depends on the sensitivity of the inferred data, and retains 92.7% context-classification accuracy. Our findings lend credence to the idea that integrating security, power management, inference, and a shared, lightweight context signal could be a great way to make ultra-low-power IoT deployments last longer and be more reliable. We also think that this approach needs more research into adversarial robustness, side-channel hardening, and long-term field testing.

References

1. Dalai, S. K., &Behera, P. (2019). Sleep scheduling protocol for data aggregation in WSN with energy optimization. In *National Conference on Innovations in Science, Technology and Management (NCISTM-2019).*
2. Goyal, T. K., Sahula, V., &Kumawat, D. (2022). Energy efficient lightweight cryptography algorithms for IoT devices. *IETE Journal of Research, 68*(3), 1722–1735.
3. Pathak, A. N., &Yadav, A. R. (2024). Scheduling based on residual energy of sensors to extend the lifetime of network in wireless sensor network. *Journal of Engineering and Applied Science, 71*, Article 100.
4. Prabhu, M., &Chandrabose, A. (2024). Optimization based energy aware scheduling in wireless sensor networks. *The Scientific Temper, 15*(Special Issue), 78–85.
5. Radhakrishnan, I., Jadon, S., &Honnavalli, P. B. (2024). Efficiency and security evaluation of lightweight cryptographic algorithms for resource-constrained IoT devices. *Sensors, 24*(12), 4008.
6. Rao, V., &Prema, K. V. (2021). A review on lightweight cryptography for Internet-of-Things based applications. *Journal of Ambient Intelligence and Humanized Computing, 12*(9), 8835–8857.
7. Singh, H. M., Agashe, S., Jain, S., Ghosh, S., Challa, A., Danda, S., &Sen, S. (2023). (POSTER) Insights from executing TinyML models on smartphones and microcontrollers. In *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)* (pp. 89–92). IEEE.
8. Sreedhar, N., Bhagya, R., &Bharthi, R. (2024). Design and implementation of Tiny ML model using STM32F platform. In *Trends in sustainable computing and machine intelligence* (pp. 169–184).