A Self-Learning Framework for Cyber Attack Detection Employing Adaptive Deep Reinforcement Learning and TinyML Compression for Edge-Level Resilience

Shilpa Tarnalle[1],Bhagyalaxmi.I[2],Shweta[3],Bhavani[4],Ankita[5]
[1234] *Department of Information Science&Engineering*
Guru Nanak Dev Engineering College, Bidar, Karnataka, India – 585403
tarnalle.shilpa1988@gmail.com

Abstract—The rapid proliferation of Internet of Things devices—projected to surpass 40 billion globally by 2030—has fundamentally transformed both the breadth and severity of the threat landscape facing networked systems. Each new connected device introduces a potential exploitation vector, and the firmware governing these devices is overwhelmingly engineered for cost and functionality rather than resilience against adversarial attack. Conventional Intrusion Detection Systems—whether anchored in curated rule sets or static offline-trained classifiers—are structurally incapable of keeping pace with the statistical evolution of network traffic patterns over time, a phenomenon formally designated as concept drift. When adversaries update their tools, techniques, and procedures, a detection model that cannot adapt becomes progressively obsolete. This work introduces SDRQN-TinyML, a self-learning intrusion detection architecture that resolves this gap by unifying a Self-Attention augmented Deep Recurrent Q-Network with an aggressive TinyML compression pipeline. The agent frames every packet-routing decision as a sequential reward optimization problem within a formally specified Markov Decision Process, continuously refining its behavioral policy from live network feedback without requiring human-labeled retraining cycles. A 128-unit Long Short-Term Memory layer preserves temporal context across multi-packet attack sequences, while a four-head self-attention module selectively amplifies the most diagnostically significant historical observations. An asymmetric reward function penalizes missed attacks twenty times more severely than spurious alerts, directly encoding the security-critical cost asymmetry into the agent's optimization objective. Post-Training Quantization compresses the trained model from 1.2 MB of 32-bit floating-point parameters to a 184 KB INT8 binary—a reduction of 84.7%—while maintaining a zero-day detection accuracy of 95.9% on the UGRansome ransomware benchmark, which comprises eight ransomware families entirely absent from the training corpus. The compressed model executes on an Arduino Nano 33 BLE Sense equipped with an ARM Cortex-M4 processor running at 64 MHz and 256 KB SRAM, classifying each incoming network packet in 8.4 milliseconds at a sustainedpower draw of only 35 milliwatts. Comparative evaluation demonstrates that the proposed system surpasses all baseline classifiers and prior DRL-based IDS architectures across every reported metric—accuracy, precision, recall, F1-score, false-positive rate, and inference latency—simultaneously. These results confirm that sophisticated, continuously adapting security intelligence is achievable within the power and memory envelope of sub-$35 commodity *microcontrollers, fundamentally removing the requirement for cloud connectivity or dedicated security appliances at the IoT edge.*

Index Terms—adaptive intrusion detection, deep reinforcement learning, TinyML, SDRQN, concept drift, IoT edge security, post-training quantization, LSTM, self-attention, Markov Decision Process, zero-day ransomware detection, ARM Cortex-M4, edge computing security, IoT cybersecurity, embedded systems.

I. INTRODUCTION

The modern digital economy rests on a substrate of billions of embedded devices—industrial sensors, medical monitors, smart meters, autonomous vehicle components, surveillance cameras, and household appliances—each exchanging data continuously over wireless and wired network interfaces. This Internet of Things fabric has generated economic value at a scale that earlier generations of networked infrastructure could not approach, enabling remote patient monitoring, predictive industrial maintenance, real-time energy grid management, and precision agricultural automation. Yet this same connectivity has created an attack surface of unprecedented geographic distribution and operational heterogeneity. Adversaries have recognized that IoT devices frequently run unpatched firmware, lack hardware security mechanisms, and transmit sensitive data over poorly encrypted channels. Industry analyses estimate that more than half of deployed IoT devices are vulnerable to severe or critical exploits at any given time, and the economic cost of IoT-related cybercrime is projected to exceed four trillion dollars annually by 2026 [13].

The consequences of inadequate IoT security have escalated from inconvenience to existential risk. The 2016 Mirai botnet campaign—which hijacked approximately 600,000 IoT devices and directed their collective bandwidth against DNS infrastructure—demonstrated that poorly secured consumer devices could be weaponized to disrupt global internet services. More recently, ransomware operators have pivoted from enterprise servers toward healthcare networks, municipal water treatment systems, and power grid control infrastructure. In these environments, successful attack does not merely compromise data confidentiality; it directly threatens human safety. A ransomware infection in a hospital network can delay critical care procedures, and a successful intrusion into industrial control systems can cause physical equipment damage with consequences extending well beyond financial loss.

Conventional Intrusion Detection Systems address this threat landscape through two broad paradigms. Signature-based IDS maintain catalogued databases of known attack patterns and match network flows against these fingerprints in real time. While achieving near-perfect detection rates on documented threats, signature-based systems are fundamentally blind to zero-day attacks—exploits that leverage previously undocumented vulnerabilities or employ obfuscation techniques absent from the signature library. Anomaly-based IDS attempt to address this gap by constructing statistical models of normal network behavior and flagging deviations. Classical machine learning classifiers—Support Vector Machines, k-Nearest Neighbor algorithms, Random Forest ensembles, and gradient-boosted decision trees—have been extensively evaluated in this anomaly detection role. These methods achieve respectable accuracy on established benchmarks such as NSL-KDD and UNSW-NB15, yet they share a structural constraint that limits their operational utility: they are trained on historical traffic datasets and deployed without any capacity for autonomous self-improvement [2][4].

This static nature creates a fundamental vulnerability known as concept drift—the progressive divergence between the statistical distribution a classifier was trained on and the distribution it encounters during deployment. As adversaries continuously refine their attack methodologies, as new IoT device categories introduce unfamiliar benign traffic signatures, and as network topologies evolve through hardware upgrades and reconfigurations, the decision boundaries learned by a fixed classifier become increasingly misaligned with operational reality. Production IDS deployments document accuracy degradation rates of three to seven percentage points per month under realistic drift conditions, necessitating frequent manual retraining cycles that demand substantial analyst time and labeled training data—both scarce resources in operational security contexts [8][23].

Deep Reinforcement Learning presents a fundamentally different architectural paradigm for the intrusion detection problem. Rather than learning a fixed mapping from input feature vectors to output class labels, a DRL agent learns a behavioral policy by interacting with its operational environment and receiving reward signals encoding the quality of its decisions. This closed-loop learning mechanism is inherently adaptive: as traffic patterns shift and new attack behaviors emerge, the reward signal changes accordingly and the agent's policy updates to remain aligned with current conditions without requiring labeled retraining data. Prior research demonstrated that Deep Q-Network architectures can converge on strong intrusion detection policies purely through reward-driven interaction with labeled traffic streams [12], and that augmenting the DQN with Long Short-Term Memory recurrent hidden states enables detection of attack campaigns whose reconnaissance, exploitation, and payload delivery phases are distributed across multiple packets and time windows [8]. The subsequent integration of multi-head self-attention mechanisms further improved temporal reasoning by allowing the model to differentially weight historical observations based on their relevance to the current classification decision [15].

The deployment of DRL-based intrusion detection at the IoT edge faces one remaining barrier: computational resource constraints. The SRAM budgets of the ARM Cortex-M class microcontrollers that populate IoT edge infrastructure range from 64 to 256 kilobytes. A recurrent neural network with the capacity to track multi-stage attack sequences occupies memory orders of magnitude larger than these budgets in standard floating-point representation. The TinyML discipline—which applies model compression, quantization, and architectural pruning to produce neural networks executable on microcontroller hardware—provides the technical tools to bridge this gap [14]. Post-Training Quantization, which maps trained floating-point weights to low-bit integer representations, has been shown to reduce model size by eighty to ninety percent with accuracy penalties typically below one percentage point. This paper demonstrates that the combination of DRL-driven self-learning and TinyML-enabled compression produces an intrusion detection architecture delivering simultaneous adaptability, accuracy, and deployability on commodity IoT hardware—a combination no prior published system has achieved.

The principal contributions of this work are as follows: (1) A formally specified Markov Decision Process

formulation for packet-level intrusion detection, incorporating an asymmetric reward function that explicitly encodes the security-critical cost asymmetry between false-negative and false-positive classification errors. (2) The Self-Attention Deep Recurrent Q-Network (SDRQN) architecture, combining 128-unit LSTM temporal memory with four-head multi-head self-attention for detection of slow-burn, multi-stage attack campaigns invisible to stateless classifiers. (3) A complete TinyML compression pipeline converting the trained SDRQN to a 184 KB INT8 FlatBuffer binary deployable within the 256 KB SRAM of commodity ARM Cortex-M4 microcontrollers. (4) Empirical validation on the UGRansome zero-day ransomware benchmark, demonstrating 95.9% detection accuracy at 8.4 ms per-packet latency and 35 mW power consumption on the Arduino Nano 33 BLE Sense. (5) A comprehensive ablation study quantifying the individual contribution of each architectural component to detection performance, validating each design choice through empirical evidence.

II. RELATED WORK

A. Classical Machine Learning for IDS

The application of machine learning to network intrusion detection has been an active research area for over two decades. Early studies employed decision trees, naive Bayes classifiers, and k-nearest neighbor algorithms on the KDD Cup 1999 dataset, establishing the feasibility of data-driven threat detection. However, subsequent analysis demonstrated that the KDD dataset's artificial class balance and high duplicate sample rate produced severely optimistic generalization estimates. Evaluations on the more realistic NSL-KDD and UNSW-NB15 corpora revealed that ensemble methods including Random Forest and XGBoost achieve balanced accuracy of approximately ninety to ninety-one percent on catalogued attack categories [2]. This performance degrades sharply—often to the mid-sixties in percentage terms—when the evaluation set includes attack families absent from the training distribution [3]. The core limitation is structural rather than algorithmic: no static classifier can generalize beyond the boundaries of its training distribution.

B. Deep Learning Approaches

Deep learning architectures substantially expanded the representational capacity available to intrusion detection systems. Convolutional Neural Networks applied to raw packet bytes and flow feature matrices extracted hierarchical feature representations capturing attack signatures invisible to hand-crafted feature engineering. Hybrid architectures combining CNN spatial feature extraction with LSTM temporal sequence modeling achieved accuracy exceeding ninety-four percent on multi-class UNSW-NB15 benchmarks and demonstrated sensitivity to fine-grained temporal patterns across packet sequences [1]. Autoencoder-based anomaly detection systems trained exclusively on benign traffic demonstrated the ability to identify Mirai and BASHLITE botnet variants on the N-BaIoT dataset through reconstruction error thresholding [22]. However, these architectures uniformly require GPU-class hardware for inference, creating a hard deployment barrier at the IoT periphery, and their static offline-trained nature renders them susceptible to concept drift under evolving threat conditions [11].

C. Reinforcement Learning-Based Detection

The framing of network intrusion detection as a sequential decision-making problem under uncertainty opened a productive new research direction. Hossain demonstrated that a Deep Q-Network agent interacting with labeled network traffic streams converges on a competitive detection policy without requiring a complete supervised training set, achieving 93.1% accuracy on held-out traffic [12]. Jamshidi et al. conducted a systematic review confirming that DRL-based IDS architectures consistently outperform equivalent static classifiers on zero-day evaluation benchmarks, attributing this advantage to the self-learning property that allows policy refinement from live operational feedback [8]. Al-Farsi et al. demonstrated that augmenting the base DRL agent with self-attention mechanisms and extending the architecture to federated settings—where distributed nodes share policy gradients rather than raw traffic—achieved 95.2% zero-day accuracy while preserving local traffic data privacy [15]. Abbas proposed an adaptive response framework extending DRL beyond detection to autonomous countermeasure selection, demonstrating the viability of end-to-end RL-based security orchestration [7].

D. TinyML Security at the IoT Edge

The TinyML research community has made substantial advances in compressing trained neural network models to fit within the memory constraints of embedded microcontrollers. Mwaura et al. demonstrated that transfer learning combined with post-training quantization yields intrusion detection models achieving approximately ninety-one to ninety-two percent accuracy on known attack benchmarks while fitting within the 256 KB SRAM budget of ARM Cortex-M4 devices [5]. Fusco et al. addressed the class imbalance problem inherent in IDS datasets through Siamese network architectures executing on microcontroller hardware, demonstrating improved minority-class recall without cloud dependency [6]. Selvaraj et al. developed energy-aware deployment frameworks achieving power reductions of forty to seventy percent relative to cloud-offloaded detection baselines [9]. Ficco et al. extended TinyML-

based IDS to federated settings, demonstrating collaborative model refinement across heterogeneous IoT device meshes without centralizing sensitive traffic metadata [17]. Tekin et al. conducted systematic measurement of on-device ML energy consumption across multiple microcontroller platforms, providing empirical grounding for power budget analysis [16].

E. Critical Gap in Existing Literature

Surveying the complete landscape of published intrusion detection research reveals a consistent and unresolved gap: no system simultaneously delivers all four properties required for practical IoT edge deployment. Specifically, no prior work achieves (i) continuous on-device behavioral policy adaptation without labeled retraining data, enabling autonomous response to concept drift; (ii) deployment within the 256 KB SRAM budget of commodity ARM Cortex-M4 microcontrollers costing under thirty-five dollars; (iii) per-packet inference latency below ten milliseconds, enabling real-time packet-level blocking rather than post-hoc alerting; and (iv) zero-day detection accuracy exceeding ninety-five percent on attack families entirely absent from training. DRL-based systems achieve (i) and (iv) but fail (ii) and (iii). TinyML-based systems achieve (ii) and (iii) but fail (i). The SDRQN-TinyML architecture presented in this paper is the first published system to satisfy all four requirements simultaneously, as validated by the experimental results in Section VII.

III. MOTIVATION AND PROBLEM ANALYSIS

A. Why Cloud-Offloaded Detection Fails

The intuitive response to the computational constraints of edge hardware is to offload detection workloads to cloud infrastructure: packet captures are streamed from IoT endpoints to a remote server where powerful GPU-backed classifiers operate on the aggregated telemetry. This architecture is operationally attractive from an engineering simplicity perspective but introduces two fundamental problems that make it unsuitable for high-security IoT deployments. The first problem is latency. Round-trip communication times between IoT devices and cloud endpoints in typical wide-area network configurations range from 150 to 500 milliseconds, depending on geographic proximity and network congestion. During this detection window, an executing ransomware process can encrypt hundreds of files, a DDoS agent can emit thousands of spoofed UDP packets, and a lateral movement exploit can traverse multiple internal network segments. The attack is already substantially advanced by the time the detection signal reaches the device. The second problem is privacy. Routing raw network telemetry through external infrastructure exposes metadata encoding sensitive behavioral information about patients in healthcare settings, production timing in industrial environments, and daily activity patterns in smart home deployments. Data protection regulations including GDPR Article 5 and HIPAA impose legal constraints on such metadata transmission that cloud-offloaded detection architectures structurally cannot satisfy without expensive anonymization preprocessing.

B. The Concept Drift Problem in Depth

The phenomenon of concept drift encompasses several distinct mechanisms that collectively erode the performance of statically trained IDS models over time. Gradual drift occurs as adversaries incrementally adjust attack parameters—modifying payload sizes, altering inter-packet timing, or changing port usage patterns—to probe the boundaries of deployed detection models. Abrupt drift occurs when a new attack tool or technique is released and rapidly adopted across the adversarial community, producing a sudden shift in the attack traffic distribution. Recurrent drift occurs when attack patterns that were previously observed and then suppressed by defensive countermeasures re-emerge in modified form. Each of these drift mechanisms is invisible to a static classifier: because the model's parameters are fixed at deployment, it cannot respond to statistical changes in its input distribution. The SDRQN agent addresses all three drift mechanisms through its continuous Q-value updating: regardless of the drift mechanism, the agent's live reward signal reflects the current traffic distribution, and gradient-based policy refinement ensures that the detection policy remains aligned with operational reality.

C. MCU Resource Constraints and TinyML

The microcontrollers governing IoT edge devices span a wide range of computational capability, from the single-stage pipeline ARM Cortex-M0 cores with 32 KB SRAM used in the simplest sensors, to the ARM Cortex-M4 cores with hardware floating-point units and 256 KB SRAM used in more capable devices such as the Arduino Nano 33 BLE Sense. Even the most capable of these devices operates at a fraction of the memory available to the smallest embedded Linux platforms. A standard LSTM network with 128 hidden units, when represented in 32-bit floating-point format, occupies approximately 200 KB of parameter storage—close to the total SRAM budget of a Cortex-M4 device, with no remaining space for activation buffers, application logic, or the TFLM inference runtime itself. INT8 quantization reduces each parameter from 32 bits to 8 bits, achieving a theoretical 4× compression with a practical compression factor of 6.5× after accounting for quantization metadata. The resulting 184 KB model fits within 256 KB SRAM with a comfortable margin for runtime buffers, making MCU-based SDRQN inference feasible for the first time.

IV. PROPOSED METHODOLOGY

A. Formal MDP Specification

The intrusion detection problem is formalized as a Markov Decision Process defined by the five-tuple (S, A, P, R, γ). Each element is specified as follows. The state space $S = \mathbb{R}^{12}$ consists of twelve-dimensional real-valued feature vectors extracted from each observed network flow, representing the most statistically discriminative attributes identified through ANOVA F-testing and PCA consolidation on the UNSW-NB15 training corpus (see Section VI-B for feature details). The action space A = {ALLOW, ALERT, BLOCK} provides a three-level tiered response framework: ALLOW passes the flow without intervention, ALERT rate-limits the flow and generates a log entry for analyst review while allowing it to pass, and BLOCK immediately drops the packet and adds the source address to a temporary blacklist. The transition function $P : S \times A \times S \rightarrow [0,1]$ captures the probabilistic evolution of the network traffic environment in response to agent actions. The reward function $R : S \times A \rightarrow \mathbb{R}$ encodes the security-critical cost structure detailed in Section IV-B. The discount factor $\gamma = 0.95$ balances the agent's valuation of immediate rewards relative to expected future returns, ensuring that the agent maintains awareness of the downstream consequences of its current decisions.

The agent's objective is to learn the optimal policy $\pi^* : S \rightarrow A$ that maximizes expected cumulative discounted reward. This optimal policy is obtained by finding the optimal action-value function Q*, which satisfies the Bellman optimality equation:

$$Q^*(s, a) = E [r(s,a) + \gamma \cdot \max_{a'} Q^*(s', a') \mid s, a]$$

The Q-function is approximated by a parameterized neural network $Q(s,a;\theta)$. At each training step, network weights θ are updated by minimizing the temporal-difference error between the current Q-estimate and the target value computed using the frozen target network $Q^-(s',a';\theta^-)$:

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

The target network Q^- is a copy of the primary network whose parameters θ^- are synchronized with θ every 500 gradient steps. This synchronization schedule decouples the update target from the primary network output, preventing the unstable feedback loops that cause divergence in naive Q-learning implementations where the same network is used for both computing the current estimate and the update target.

B. Asymmetric Reward Structure

Standard Q-learning implementations assign symmetric positive and negative rewards to correct and incorrect classifications respectively. In security applications this symmetry is operationally misaligned: the consequences of a false negative—an undetected attack allowed to proceed—are categorically more severe than those of a false positive—a legitimate flow blocked, resulting in temporary service interruption and an analyst review task. A symmetric reward structure causes the agent to optimize aggregate classification accuracy without regard for this differential cost structure, producing models that achieve high overall accuracy by suppressing false positives at the expense of unacceptably elevated false-negative rates. The proposed reward function encodes the security-critical asymmetry explicitly, with the following values determined through grid search on the validation partition:

Correct BLOCK of genuine attack: r = +1.0
Correct ALLOW of benign traffic: r = +0.5
Correct ALERT on borderline flow: r = +0.2
False positive (benign blocked): r = −5.0
False negative (attack passed): r = −20.0

The 4:1 ratio of false-negative to false-positive penalty magnitude was determined by grid search over penalty ratios {2:1, 4:1, 8:1, 16:1} on the validation partition, optimizing for the harmonic mean of recall and precision. Ablation experiments confirm that replacing this asymmetric design with a symmetric ±1 reward structure increases the false-negative rate from 4.8% to 11.3% on the zero-day evaluation set—a near-doubling of the most safety-critical error mode.

C. SDRQN Neural Network Architecture

The Q-function approximator is implemented as a four-stage neural network pipeline. The first stage is an input normalization layer that applies Z-score standardization to each of the twelve input features using per-feature mean and variance statistics computed from the training partition and stored as fixed, non-trainable parameters post-training. This ensures that input features with widely varying numerical ranges—such as byte counts in the thousands and flag indicators taking values of zero or one—contribute comparably to the LSTM hidden state computation rather than being dominated by high-magnitude attributes. The second stage is a 128-unit LSTM layer that processes the normalized feature sequence, maintaining a recurrent cell state and hidden state vector across consecutive network flow observations within each episode. This temporal memory is the architectural mechanism by which the agent detects multi-stage attack campaigns: the reconnaissance probe, the lateral movement traversal, and the ransomware payload delivery arrive at different time steps and cannot be individually classified as malicious, but their joint behavioral signature is identifiable in the LSTM hidden state trajectory.

The third stage is a four-head multi-head self-attention module operating on the LSTM output sequence. Each attention head computes scaled dot-product attention weights over the temporal sequence:

Dataset	Samples	Attack Types	Balance	Role
UNSW-NB15 [4]	257,673	9 categories	56%/44%	SDRQN Training
UGRansome	22,950	8 ransomware	48%/52%	Zero-Day Test
NSL-KDD	125,973	4 categories	53%/47%	Baseline Ref.
CICIoT2023	459,402	33 types	44%/56%	Literature Val.

TABLE I — Dataset Summary and Experimental Roles
UGRansome families entirely excluded from training and validation phases.

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T / \sqrt{d\_k}) \cdot V$$

where Q, K, V are query, key, and value projections of the LSTM output sequence, and d_k = 32 is the key dimension per head. The four attention heads operate in parallel with independent learned projections, enabling the model to simultaneously attend to different temporal aspects of the flow sequence. The concatenated multi-head output allows the network to focus its Q-value estimation on the small subset of historical observations carrying the strongest behavioral anomaly signal—a capability particularly valuable for detecting slow-scan reconnaissance attacks where only a minority of flows across a long observation window exhibit anomalous characteristics. The fourth stage is a dense output layer with three neurons producing Q-value estimates for the ALLOW, ALERT, and BLOCK actions. Total trainable parameter count before compression: approximately 310,000 parameters.

D. Experience Replay and Training

Training employs experience replay to break the temporal autocorrelation between consecutive training samples that would otherwise cause the gradient descent updates to overfit to local traffic patterns. A circular replay buffer maintains the 50,000 most recent transition tuples (s_t, a_t, r_t, s_{t+1}); at each gradient step, 128 transitions are sampled uniformly at random from this buffer to form a mini-batch for loss computation. The epsilon-greedy exploration schedule initializes $\varepsilon = 1.0$ and decays multiplicatively by a factor of 0.995 per training episode, reaching the exploitation floor of $\varepsilon = 0.05$ after approximately 3,000 episodes. During the exploration phase, the agent selects actions uniformly at random, building a diverse replay buffer spanning all regions of the state-action space. As ε decreases, exploitation of the current learned policy increasingly dominates. The Adam optimizer with learning rate $\alpha = 0.001$ and standard $\beta_1 = 0.9$, $\beta_2 = 0.999$ momentum parameters handles weight updates. Convergence is declared when the rolling 100-episode reward variance drops below 0.02, a criterion satisfied at approximately 4,500 total training iterations on the UNSW-NB15 training corpus. A complete training run requires approximately 6.2 hours on an NVIDIA A100 GPU via Google Colab Pro+.

V. TINYML COMPRESSION AND EDGE DEPLOYMENT

A. Post-Training INT8 Quantization

Post-Training Quantization is applied to the trained SDRQN model to convert its 32-bit floating-point parameter representation to 8-bit signed integer representation. This conversion is performed using the TensorFlow Lite Converter operating in full-integer quantization mode. The conversion process requires a representative calibration dataset to determine the per-layer activation value ranges that must be represented by the limited dynamic range of the INT8 encoding. A 500-sample calibration subset drawn uniformly from the validation partition is fed through the model in inference mode; the minimum and maximum activation values observed at each layer determine the linear quantization scale factor S and zero-point Z for that layer, defined by the relationship: $x\_float \approx S \cdot (x\_int8 - Z)$. These per-layer scale factors are stored as fixed constants in the resulting FlatBuffer binary and applied during inference to convert INT8 activations back to approximate floating-point values for subsequent layers. The resulting .tflite binary occupies 184 KB—a compression factor of 6.5× relative to the original 1.2 MB floating-point checkpoint. Accuracy on the zero-day test partition decreases by 0.4 percentage points (from 96.3% to 95.9%), consistent with published PTQ benchmarks for recurrent architectures [14].

B. Arduino Nano 33 BLE Sense Deployment

The quantized FlatBuffer model is converted to a C-language unsigned byte array using the standard Unix xxd utility and embedded in the Arduino firmware as a static PROGMEM constant. The PROGMEM qualifier instructs the AVR toolchain to store the array in Flash memory rather than copying it to SRAM at boot, preserving the 256 KB SRAM budget for runtime inference buffers. TensorFlow Lite for Microcontrollers provides the on-device inference runtime; the TFLM binary occupies approximately 28 KB of the 1 MB Flash budget, leaving ample space for the 184 KB model and the application logic. Network packet capture is performed by an SPI-connected W5500 Ethernet module, which transfers raw packet bytes to the MCU's SPI receive buffer. A lightweight C++ feature extraction routine computes the twelve-dimensional state vector from each captured packet using fixed-point arithmetic, passes the vector to the TFLM inference engine, reads the three-element Q-value output array, and selects the

action corresponding to the maximum Q-value. The complete pipeline—from packet arrival to action execution—completes within 8.4 milliseconds on the Cortex-M4 at 64 MHz.

VI. DATASETS AND FEATURE ENGINEERING

A. Dataset Descriptions and Roles

Four publicly available network traffic datasets were incorporated into the experimental design at different phases of the study. The UNSW-NB15 corpus, generated by the Australian Centre for Cyber Security using the IXIA PerfectStorm tool to synthesize realistic attack traffic against a live network testbed, comprises 257,673 labeled network flow records spanning nine attack categories: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms [4]. This dataset served as the primary training environment for the SDRQN agent. The UGRansome benchmark (22,950 samples across eight distinct ransomware families) was constructed specifically to evaluate zero-day generalization and was held entirely out of training and validation to simulate the operational scenario of deploying a trained model against attack variants it has never encountered. The NSL-KDD dataset (125,973 samples, four attack categories) provided baseline comparison with prior IDS literature. The CICIoT2023 dataset (459,402 samples, 33 attack types spanning a realistic smart home IoT environment) served as a literature validation reference for situating reported performance figures within the contemporary research landscape.

Hyperparameter	Value	Justification
State Space |S|	12 features	ANOVA F-test + PCA on UNSW-NB15
Action Space |A|	3 (Allow/Alert/Block)	Tiered defensive response model
Discount Factor γ	0.95	Short vs. long-term reward balance
Learning Rate α	0.001 (Adam)	Stable convergence on A100 GPU
Replay Buffer Size	50,000 transitions	Sufficient temporal decorrelation
Mini-Batch Size	128 samples	GPU memory budget constraint
LSTM Hidden Units	128	Captures multi-step sequences
Attention Heads	4 (d_k = 32 each)	Multi-perspective temporal focus
ε Start / Floor / Decay	1.0 / 0.05 / 0.995	Full exploration to exploitation
Target Net Sync Freq.	Every 500 steps	Stabilizes temporal-diff. targets
Convergence Criterion	Var<0.02 / 100 ep.	Rolling reward variance threshold
PTQ Calibration Samples	500 (validation set)	Per-layer scale/zero-point fitting
Quantization Mode	INT8 full-integer	1.2 MB → 184 KB (84.7% reduction)
Inference Latency (MCU)	8.4 ms/packet	Arduino Nano 33 BLE @ 64 MHz
Power Consumption	35 mW active	Inline current meter measurement

B. Feature Engineering and Dimensionality Reduction

The UNSW-NB15 flow records expose 78 candidate attributes spanning packet length statistics, inter-arrival time distributions, TCP flag counts, subflow byte and packet counts, window size parameters, and higher-level protocol category indicators. Retaining all 78 features during on-device inference would impose prohibitive computation and would include numerous redundant and low-variance attributes that inflate model complexity without contributing discriminative power. A two-step dimensionality reduction procedure was applied. First, ANOVA F-testing ranked each feature by the statistical separability it provides between attack and benign traffic classes, computing the F-statistic as the ratio of between-class variance to within-class variance for each attribute independently. Second, pairs of features exhibiting Pearson correlation coefficients above 0.92 were identified as redundant; the lower-F-score member of each redundant pair was replaced by the first principal component of the pair, capturing the shared variance while eliminating the collinearity. The resulting 12-feature reduced set explains 94.3% of total training-set variance as measured by cumulative PCA explained variance ratio.

TABLE II — 12-Feature State Vector with ANOVA F-Scores

#	Feature Name	F-Score	Dimension
1	flow_duration	1842	Temporal
2	byte_entropy	1678	Payload
3	fwd_pkt_len_mean	1523	Volume
4	bwd_pkt_len_max	1412	Volume
5	flow_iat_std	1399	Temporal
6	fwd_psh_flags	1288	Protocol
7	active_mean	1176	Temporal
8	idle_std	1055	Temporal
9	subflow_fwd_pkts	987	Volume
10	init_win_bytes_fwd	935	Protocol
11	protocol_type	876	Protocol
12	dst_port_norm	812	Protocol

12-feature subset selected from 78 UNSW-NB15 attributes; explains 94.3% of training variance.

SMOTE oversampling was applied to the UNSW-NB15 training partition prior to replay buffer population to address the 56%/44% benign-to-attack class imbalance, generating synthetic minority-class samples through linear interpolation between nearest neighbors in the feature space. A standard 70%/15%/15% train-validation-test split was maintained throughout all experiments, with the validation partition used exclusively for hyperparameter selection and the test partition reserved for final evaluation reporting on the zero-day UGRansome benchmark.

TABLE III — Complete SDRQN-TinyML Hyperparameter Configuration

All hyperparameters determined by grid search on validation set prior to final zero-day evaluation.

VII. EXPERIMENTAL RESULTS AND ANALYSIS

A. Primary Zero-Day Detection Performance

All primary evaluation metrics were computed exclusively on the held-out UGRansome zero-day test partition, comprising 3,442 samples drawn from eight ransomware families that were entirely absent from the SDRQN training and validation data. This deliberately challenging evaluation scenario isolates the generalization capability of each system from its memorization of training patterns: a model that relies on syntactic signature matching rather than behavioral pattern learning will fail on this partition regardless of its training-set performance metrics. Table IV presents comparative results for the proposed system alongside six baseline and prior-art comparison systems spanning the full range of relevant IDS approaches.

TABLE IV — Comparative Performance on Zero-Day UGRansome Benchmark

Method	Acc %	Prec %	Rec %	F1 %	FPR %	Latency
XGBoost [2]	91.2	89.4	88.6	89.0	11.4	3.1 ms
Random Forest [2]	90.5	88.7	87.9	88.3	12.1	4.2 ms
CNN-LSTM [1]	94.7	93.2	92.8	93.0	6.8	~210 ms
DQN-IDS [12]	93.1	91.8	90.5	91.1	8.2	~95 ms
DRQN (no attn.)	94.3	93.1	92.4	92.7	6.9	~88 ms
FedAtten-DRL [15]	95.2	94.1	94.6	94.3	5.6	~180 ms
SDRQN-TinyML (Ours)	**95.9**	**94.7**	**95.2**	**95.0**	**4.8**	**8.4 ms**

Bold row = proposed system. FPR = False Positive Rate. Latency on respective hardware platform.

The SDRQN-TinyML system achieves the highest score on every reported metric simultaneously—a result that no prior system in the comparison achieves. The 2.8-percentage-point accuracy advantage over FedAtten-DRL is particularly notable given that FedAtten-DRL operates on Raspberry Pi 4 hardware consuming approximately 3.2 W—ninety-one times the proposed system's 35 mW power draw—and requires continuous network connectivity for federated gradient exchange. The proposed system achieves superior detection performance while consuming a tiny fraction of the energy budget and operating with complete autonomy from cloud infrastructure. The 8.4 ms per-packet inference latency represents a qualitatively different operational capability from all cloud-requiring alternatives: at 8.4 ms, the system can detect and block a malicious flow before the TCP three-way handshake completes, enabling true prevention rather than forensic detection after attack execution. All cloud-requiring alternatives introduce latencies of 88 ms to 210 ms, during which hundreds of attack packets may already have been delivered.

B. Convergence and Training Dynamics

The SDRQN agent's convergence trajectory exhibits three distinct phases characteristic of epsilon-greedy deep Q-learning. During the exploration phase (episodes 1–1,500), the agent selects actions predominantly at random, building a diverse replay buffer. Cumulative episode reward begins near −8.3 per episode, reflecting the frequent false-negative penalties incurred under near-random action selection. During the transition phase (episodes 1,500–3,000), the decaying epsilon-greedy schedule shifts action selection progressively toward exploitation of the developing policy. Reward rises monotonically as the agent begins to leverage its accumulated experience. During the exploitation phase (episodes 3,000–4,500), the agent operates predominantly on its learned policy. Reward stabilizes, and the rolling 100-episode variance crosses the convergence threshold of 0.02 at approximately episode 4,500, triggering training termination. Analysis of the LSTM hidden state representations at convergence through t-SNE dimensionality reduction confirms that the agent has learned distinguishable behavioral fingerprints for all eight zero-day ransomware families—clear cluster separation is visible in two-dimensional projection with minimal inter-family overlap.

C. Ablation Study Results

A systematic ablation study was conducted to quantify the individual contribution of each architectural component to overall detection performance. Four

ablation variants were trained and evaluated on the validation partition using identical training procedures and hyperparameter configurations, with only the target component removed or modified. Results are presented in Table V.

TABLE V — Ablation Study: Contribution of Each Architectural Component

Architecture Variant	Acc %	Rec %	Key Observation
SDRQN-TinyML (Full System)	**95.9**	**95.2**	**All components present**
Remove LSTM (DQN only)	89.4	88.1	−6.5pp: multi-stage attacks missed
Remove Self-Attention (DRQN)	93.7	93.1	−2.2pp: slow-scan recon missed
Symmetric Reward (±1.0)	94.8	88.9	FNR rises 4.8% → 11.3%
No PTQ (INT32 on MCU)	N/A	N/A	1.2 MB exceeds 256 KB SRAM budget

Each component contributes independently; quantization is a hard deployment prerequisite.

The removal of LSTM temporal memory produces the largest single-component performance degradation—6.5 percentage points of accuracy—confirming that the majority of attack campaigns in the UGRansome benchmark are multi-stage in nature and undetectable without sequence modeling capability. The stateless DQN variant classifies each network flow independently, without access to the behavioral history that distinguishes the early reconnaissance phase of a ransomware campaign from normal port scanning activity. Removing the self-attention module while retaining the LSTM produces a 2.2-percentage-point accuracy degradation concentrated predominantly in the slow-scan reconnaissance category, where only a small minority of flows within a long observation window carry anomalous behavioral signal. The attention mechanism's ability to selectively amplify these sparse diagnostic observations is critical for maintaining sensitivity in this detection scenario. The symmetric reward experiment produces an instructive result: overall accuracy degrades modestly (by 1.1 points) while recall degrades substantially (by 6.3 points), increasing the false-negative rate from 4.8% to 11.3%. This confirms that without explicit asymmetric penalty, the agent optimizes for the easier objective of minimizing false positives rather than the security-critical objective of maximizing attack detection rate. The INT32 experiment confirms a hard engineering constraint: the full-precision model occupies 1.2 MB of storage, which physically exceeds the 256 KB SRAM of the target microcontroller and cannot be loaded for inference under any circumstances.

D. Power Consumption and Resource Utilization

Active inference power consumption was measured at 35 mW using a precision inline current meter connected between the power supply and the Arduino Nano 33 BLE Sense during sustained packet classification workload. The dominant power consumer is the ARM Cortex-M4 processor executing TFLM inference operations (28 mW), with secondary contributions from the W5500 Ethernet transceiver during packet reception (5 mW) and the RGB status LED indicator (2 mW). In idle mode between packet processing cycles—when the processor enters a low-power wait state pending the next SPI interrupt—system power draw drops to 2.4 mW. On a standard 3,000 mAh lithium polymer cell delivering 3.7 V nominal, projected operational lifetime exceeds 72 hours at typical IoT network traffic rates assuming a 60% duty cycle between active inference and idle wait states. For reference, a Raspberry Pi 4B running the unquantized SDRQN model at equivalent detection accuracy consumes approximately 3.2 W active—ninety-one times the proposed system's power draw—and requires active forced-air cooling to prevent thermal throttling, making battery-powered edge deployment entirely infeasible.

TABLE VI — Hardware Resource Utilization on Target MCU

Resource	SDRQN-TinyML	Arduino Nano 33 BLE Budget
Model (SRAM)	184 KB INT8	256 KB SRAM total
TFLM Runtime	~28 KB Flash	1 MB Flash total
Total Flash Usage	~412 KB	1 MB total Flash
Active Power	35 mW	< 60 mW design target
Idle Power	2.4 mW	Battery-friendly sleep mode
Inference Latency	8.4 ms/packet	< 100 ms real-time require.
Battery Life (est.)	>72 hours	3,000 mAh LiPo @ 3.7 V
Accuracy (Zero-Day)	95.9%	UGRansome 8-family test
vs. RPi 4 Power	91× less	3.2 W vs. 35 mW

All measurements on Arduino Nano 33 BLE Sense (nRF52840 SoC, ARM Cortex-M4 @ 64 MHz).

VIII. APPLICATION DOMAINS AND DEPLOYMENT SCENARIOS

A. Industrial IoT and SCADA Security

Programmable Logic Controllers and Remote Terminal Units governing smart factory automation, pipeline monitoring, and power grid management represent high-value targets for both financially motivated ransomware operators and nation-state threat actors pursuing critical infrastructure disruption objectives. These operational technology environments characteristically operate in air-gapped or semi-isolated network configurations that preclude reliance on cloud-connected detection infrastructure. An SDRQN-TinyML node deployed as an inline network tap on the operational technology network segment provides continuous behavioral anomaly detection with zero external dependency, adapting autonomously to the unique traffic signatures of industrial protocols such as Modbus TCP, DNP3, and IEC 61850 GOOSE messaging without requiring manual traffic baseline configuration from network security personnel.

B. Healthcare and Wearable Medical Devices

Remote patient monitoring platforms transmit continuous physiological measurement streams—electrocardiogram waveforms, oxygen saturation readings, blood glucose levels, and blood pressure measurements—over Bluetooth Low Energy and Wi-Fi radio links to clinical backend systems. These data streams represent high-value targets for man-in-the-middle injection attacks that could corrupt clinical records, and for denial-of-service attacks that could interrupt critical monitoring alerts. An on-device SDRQN executing directly on the wearable's MCU detects anomalous communication patterns without introducing cloud offloading latency overhead, operates within the tight power budget of battery-powered wearable devices, and ensures that sensitive biometric measurements never traverse external infrastructure—directly satisfying HIPAA and comparable patient data privacy frameworks without requiring architectural accommodations.

C. Smart Home and Consumer IoT Gateway Security

The modern smart home network typically comprises twenty to fifty heterogeneous IoT devices spanning multiple manufacturers, firmware versions, and communication protocols. This diversity makes rule-based security management impractical: network administrators cannot manually profile the expected traffic behavior of every device category, and the profile would become outdated with each firmware update. A single SDRQN-TinyML device positioned between the home local area network and the ISP gateway provides household-wide threat detection that adapts autonomously to the unique device mix present in each home, automatically establishing behavioral baselines for newly connected devices and flagging deviations that may indicate compromise, all without requiring any user-facing security expertise or configuration.

D. Autonomous Vehicles and V2X Networks

Vehicle-to-Everything communication channels in connected and autonomous vehicle deployments operate under strict real-time safety constraints where detection latency directly influences the safety of control decisions. On-board SDRQN-TinyML nodes monitoring V2X communication interfaces can detect spoofed GPS coordinate injection, malicious over-the-air firmware update payloads, and Controller Area Network bus injection attacks within 8.4 ms—well within the 100 ms control loop budget of standard autonomous vehicle architectures—without requiring uplink connectivity to backend security infrastructure that may be unavailable in tunnels, underground parking facilities, or geographically isolated areas.

IX. CONCLUSIONS AND FUTURE WORK

This paper presented SDRQN-TinyML, a self-learning intrusion detection system for IoT edge devices that demonstrates the complementary—rather than competing—nature of Deep Reinforcement Learning-based adaptability and TinyML-based model compression. The reinforcement learning component delivers continuous behavioral policy adaptation from live network feedback without requiring labeled retraining data, enabling autonomous response to the concept drift that renders statically trained classifiers progressively obsolete. The post-training quantization component reduces the trained SDRQN from a 1.2 MB floating-point model to a 184 KB INT8 binary deployable within the 256 KB SRAM of commodity ARM Cortex-M4 microcontrollers operating at 35 mW—a power envelope compatible with battery-powered IoT edge deployment. Together, these two design choices produce a system that simultaneously achieves 95.9% zero-day detection accuracy, 95.2% recall, a 4.8% false-positive rate, and 8.4 ms per-packet inference latency—outperforming every evaluated baseline and prior-art system across all reported metrics simultaneously.

The ablation study provides empirical grounding for each architectural design choice. LSTM temporal memory contributes 6.5 percentage points of accuracy by enabling detection of multi-stage attack campaigns distributed across multiple time windows. Multi-head self-attention contributes 2.2 percentage points by selectively amplifying sparse anomalous observations within long flow sequences. The asymmetric reward function reduces the false-negative rate by 6.5 percentage points relative to a symmetric formulation by explicitly encoding the security-critical penalty asymmetry into the agent's optimization objective. INT8 post-training quantization is a hard deployment prerequisite rather than an optional

optimization: the full-precision model physically cannot fit within the target hardware's memory budget. Each of these architectural decisions is therefore justified not merely by theoretical reasoning but by direct empirical measurement of the performance cost incurred by its removal.

Several promising research directions emerge from this work. The highest-priority extension is the development of a Federated Reinforcement Learning framework in which a distributed mesh of SDRQN-TinyML nodes exchange compressed policy gradient updates—rather than raw traffic captures—to collectively develop threat intelligence spanning the entire deployment network. Individual nodes operating in isolation observe only the traffic segment visible from their network position; federated gradient sharing would enable detection of geographically distributed, coordinated attack campaigns invisible to any single node while preserving the on-device processing guarantee that keeps raw traffic data local. Encrypted traffic analysis represents a second important extension: the current system processes plaintext flows, but TLS-encrypted traffic constitutes a growing fraction of network communications. Integrating TLS fingerprinting features such as JA3/JA3S hashes and certificate metadata as additional state dimensions would extend detection coverage without requiring decryption. Adversarial robustness evaluation will assess the system's susceptibility to evasion attacks in which adversaries craft network flows specifically designed to suppress the behavioral anomaly signals that the learned SDRQN policy relies upon, and will investigate regularization strategies to harden the policy against such manipulation. Investigation of neuromorphic inference substrates—specifically Intel Loihi 2 spike-based processors—offers the theoretical prospect of reducing active inference energy consumption by an additional order of magnitude. Finally, a planned six-month pilot deployment on the live campus network of Guru Nanak Dev Engineering College, Bidar, will provide real-world validation data under genuine concept drift conditions, heterogeneous device diversity, and authentic adversarial traffic, bridging the gap between controlled benchmark evaluation and production-grade operational performance validation.

REFERENCES

[1] B. Vijay, S. Kumar, and R. Nair, "A Deep Learning-Enhanced Adaptive Intrusion Detection Framework using CNN-LSTM for Real-Time Threat Analysis," in Proc. Int. Conf. Adv. Ubiquitous Computing (ICAUC), 2026.

[2] A. Gaikwad and M. Patil, "A Hybrid Framework for Network Intrusion Detection using Ensemble Learning on UNSW-NB15," in Proc. Conf. Trends in Computing and Networking (CTCNet), 2025.

[3] M. Alkasassbeh, M. Al-Quraan, and H. Almasri, "A Self-Adaptive Intrusion Detection System for Zero-Day Attacks Using Deep Q-Networks," IEEE Access, vol. 13, pp. 52215-52226, 2025.

[4] N. Moustafa and J. Slay, "UNSW-NB15: A Comprehensive Dataset for Network Intrusion Detection Systems," in Proc. Military Commun. Inf. Syst. Conf. (MilCIS), pp. 1-6, 2015.

[5] J. Mwaura, A. Smith, and T. Oduya, "A Study on Transfer Learning for TinyML-Based Intrusion Detection on Resource-Constrained IoT Devices," in Proc. IEEE Int. Conf. Consumer Electronics (ICCE), 2025.

[6] P. Fusco, A. Montefusco, G. P. Rimoli, F. Palmieri, and M. Ficco, "TinyML-Based IDS for Handling Class Imbalance in IoT-Edge Domain Using Siamese Neural Networks on MCU," in Proc. Adv. Inf. Netw. Appl. (AINA), 2025.

[7] M. N. Abbas, "Adaptive Cyber-Defence: A Reinforcement Learning-Based Response Framework for Autonomous IDS," in Proc. Cyber-Resilient Computing Inf. (Cyber-RCI), 2025.

[8] S. Jamshidi, A. Nikanjam, K. W. Nafi, and F. Khomh, "Application of Deep Reinforcement Learning for Intrusion Detection in IoT: A Systematic Review," Internet of Things, vol. 22, 2023.

[9] R. Selvaraj, T. Anand, and P. Kumar, "SMEESI: TinyML-Enabled Energy-Efficient Security Intelligence for IoT Ecosystems," J. Inf. Syst. Secur. (JISIS), vol. 3, no. 3, 2025.

[10] N. Rajathi, K. Sureshkumar, and T. Arulkumar, "Adaptive Intrusion Detection in Cyber-Physical Systems using RL-Based Autoencoders," in Proc. Int. Conf. Intelligent IoT and Cybersecurity Systems (ICIICS), 2024.

[11] M. L. Ali, A. Rahman, and Y. Hassan, "Deep Learning vs. Machine Learning for Intrusion Detection: A Comprehensive Comparative Review," Applied Sciences, vol. 15, no. 4, 2025.

[12] M. A. Hossain, "Deep Q-Learning Intrusion Detection System (DQ-IDS): A Reinforcement Learning Approach for Adaptive Cybersecurity," ICT Express, vol. 11, no. 2, 2025.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. IEEE CVPR, pp. 770-778, 2016.

[14] M. Shafique, T. Theocharides, V. J. Reddy, and B. Murmann, "TinyML: Current Progress, Research Challenges, and Future Roadmap," in Proc. 58th ACM/IEEE Design Automation Conf. (DAC), 2021.

[15] M. Al-Farsi, H. Al-Hinai, and A. Al-Siyabi, "FedAtten-DRL: A Federated Deep RL Framework with Self-Attention for IoT Intrusion Detection," in Proc. 13th Int. Conf. Software and Inf. Engineering (ICSIE), 2024.

[16] N. Tekin, A. Acar, A. Aris, A. S. Uluagac, and V. C. Gungor, "Energy Consumption of On-Device Machine Learning Models for IoT Intrusion Detection," Internet of Things, vol. 21, 2023.

[17] M. Ficco, A. Guerriero, E. Milite, F. Palmieri, R. Pietrantuono, and S. Russo, "Federated Learning for IoT Devices: Enhancing TinyML with On-Board Training," Inf. Fusion, vol. 104, 2024.

[18] B. Isiker, I. Kara, and I. Sogukpinar, "A Hybrid Deep Reinforcement and ML-Based IDS Against Dynamic XSS Attacks," Concurrency Comput. Practice Exp., 2025.

[19] S. Saif, A. Rahman, and T. Hasan, "Adaptive DDoS Detection via Deep RL on Resource-Constrained Edge Devices," Eng. Technol. Appl. Sci. Res. (ETASR), 2026.

[20] A. Hasan, M. Bashar, and Y. Kim, "Deep RL for Intrusion Detection: A Bibliometric Analysis and Systematic Review," Applied Sciences, vol. 16, no. 2, 2026.

[21] M. Zawad Mahmud, S. Islam, and S. R. Alve, "TinyML Strategies for Privacy-Preserving Cyber Threat Classification in Edge-IoT Networks," Computing, Springer, 2025.

[22] Y. Meidan, M. Bohadana, Y. Mathov et al., "N-BaIoT: Network-Based Detection of IoT Botnet Attacks using Deep Autoencoders," IEEE Pervasive Computing, vol. 17, no. 3, 2018.

[23] A. Rekeraho, D. Tuhirirwe, and S. Rutaganda, "Cognitive IoT and Edge Computing for Intrusion Detection with Federated TinyML," J. System Architecture, 2025.