LighTellite: Reinforcement Learning-Based Framework for Energy Efficient Onboard Satellite Anomaly Detection

Aviel Ben Siman Tov, Edita Grolman, Yuval Elovici, Asaf Shabtai

Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Israel
{avielben, edita}@post.bgu.ac.il, {elovici, shabtaia}@bgu.ac.il

***Abstract*—Satellites' stable operation relies on anomaly detection (AD), which is used to identify abnormal behavior in onboard systems. However, traditional AD methods struggle to function effectively in the resource-constrained environment of satellites, where energy, memory, and computation are severely limited. This challenge is especially evident in CubeSats, the most widely deployed class of small satellites, where such constraints limit the applicability of conventional AD methods and lead to a degradation in overall performance. We introduce LighTellite, a reinforcement learning-based dual-agent framework that aims to balance AD performance and energy efficiency, in which one agent determines energy budgets, and the other dynamically selects the optimal model among a pretrained pool of AD models (each with different performance and energy characteristics). LighTellite's dynamic AD model selection enables context-aware adaptation in response to both onboard satellite data and available resources, resulting in an improvement in AD performance while maintaining low energy consumption. Experiments conducted on AegisSat, a state-of-the-art CubeSat testbed, show that our proposed framework improved attack detection rate by 10% while reducing inference energy consumption by 21.8% compared to the best static AD models (in which the same model is used throughout the entire orbit). The code and additional materials are available in the GitHub repository.**

I. Introduction

Anomaly detection (AD) plays a vital role in uncovering unexpected or malicious behavior in complex systems, especially when explicit anomaly labels are unavailable [1], [2]. By learning the patterns associated with normal operation, AD methods can highlight deviations indicative of failures or cyberattacks. The importance of AD increases in domains in which system reliability is critical and failures can have serious consequences [3]; Satellites, and particularly CubeSats, represent such a domain. CubeSats' low cost and accessibility make them ideal for a wide range of missions [4], however their strict limitations on energy and computing power increase their exposure to operational risks [5], [6].

Workshop on Security of Space and Satellite Systems (SpaceSec) 2026
23 February 2026, San Diego, CA, USA
ISBN 978-1-970672-02-2
https://dx.doi.org/10.14722/spacesec.2026.23013
www.ndss-symposium.org

Although a few AD methods using deep learning architectures such as long short-term memory (LSTM) and autoencoders (AEs) [7], [8], [9], [10] have been designed specifically for satellites, they typically have high computational and energy costs [11]. To address these limitations, we propose LighTellite, a reinforcement learning (RL)-based framework for onboard AD that balances the trade-off between AD performance with energy efficiency. Our primary goal is to enable reliable and efficient AD, while ensuring that during the satellite's routine operations, the AD do not lead to excessive energy consumption. Instead of relying on a single fixed AD model that yields the best detection capabilities (without energy consideration), LighTellite employs two cooperating agents that, for every time window, select an AD model from a set of available models to detect anomalies while considering energy constraints; performing dynamic model selection (DMS). One agent is responsible for determining the available energy for AD, while the other agent selects an ideal AD model to detect anomalies in recent satellite data.

LighTellite was evaluated using the AegisSat [12] testbed, a high-fidelity CubeSat simulation platform that integrates both hardware and software components and allows the implementation and evaluation of cyberattacks. In 58 complete orbital simulations, we reproduced realistic operational conditions, with a subset of runs deliberately injected with cyberattacks consisting of excessive resource usage (CPU, memory, and I/O operations) and data leakage through radio frequency (RF) communication, or both. The remaining simulations represented benign, attack-free operation and served as the benign baseline training. Results show that LighTellite improved attack detection rates by 10% while reducing inference energy consumption by 21.8% compared to the best static AD models, enhancing the detection capabilities while operating close to the most possible lightweight energy consumption bound.
In summary, our contributions are as follows:

- To the best of our knowledge, we are the first to design an energy-aware AD framework for satellites (demonstrated on CubeSats) that operates under strict energy constraints.
- To the best of our knowledge, while RL-based DMS exists and has been used in other domains [13], [14], [15], [16], [17], [18], our work is among the first to employ RL for DMS specifically in the satellite domain.

- We consider energy efficiency as a main objective, while existing AD approaches in the space domain focus primarily on improving detection performance.
- We provide an open-source dataset of CubeSat data, collected via the AegisSat testbed, consisting of 58 full-orbit simulations (23 benign and 35 with labeled attacks, including both single and hybrid attack scenarios).

II. Background

A. Satellites

Satellites are spacecraft that enable services ranging from phone and TV links to global navigation and Earth monitoring [19], [20]. Satellites can be classified into three orbital categories [21]: 1) Low-Earth orbit (LEO) satellites operate at altitudes between 160 and 2,000 kilometers, with an orbital period of around 90-120 minutes, and are commonly used for applications requiring low signal latency and high-resolution imaging; 2) Medium-Earth orbit (MEO) satellites, positioned at altitudes between 2,000 and 35,786 kilometers, complete an orbit approximately every 12 hours and are primarily used for navigation and communication systems; 3) Geostationary orbit (GEO) satellites are located at a constant altitude of 35,786 kilometers and orbit the Earth at the same as Earth's rotational speed, which allows them to remain fixed relative to a point on the Earth's surface, making them suitable for continuous telecommunication and meteorological monitoring [21]. All three categories are required to operate in harsh space environments that include limited resources, continuous radiation, and extreme thermal changes [22], [23].

These operational constraints are even more pronounced in resource-limited spacecraft such as CubeSats, which are miniature satellites designed for low-cost access to space that are widely deployed in LEO missions [5], [24]. Built from standardized one-unit (1U) modules measuring 10×10×10 centimeters [25], CubeSats must perform all payload, processing, and communication tasks with an average energy budget of only 1 to 2.5 watts, a budget that is dictated by the small surface area of their solar panels [26]. As launch costs have decreased, the number of CubeSat missions has grown substantially and they are now deployed for a wide range of purposes. Their widespread adoption in a variety of domains, including remote sensing, military surveillance, and education, highlights the importance of efficient management of onboard resources to ensure reliable operation [4].

Given their tight resource constraints, CubeSats are particularly vulnerable to cyber threats that exploit their limited energy and processing capabilities [27]; For example, denial-of-service (DoS) attacks, such as resource exhaustion or traffic flooding, which can drain onboard energy reserves, can potentially lead to mission degradation or total failure [28]. The risk is further heightened during eclipse phases [29], when CubeSats operate solely on battery and cannot replenish energy through solar charging.

B. Reinforcement Learning

RL is a machine learning (ML) paradigm in which an agent interacts with an environment to learn how to make better decisions based on collected feedback signals called rewards [30], [31]. Unlike unsupervised learning, which extracts patterns from unlabeled data, and supervised learning, which fits models to labeled data, RL has no predefined correct outputs; the agent must discover, by trial and error, which actions maximize rewards in each state it encounters.

In each timestep, the agent observes the environment's state, selects an action, and receives the resulting next state, along with an immediate reward. This interaction between the agent and the environment is commonly modeled as a Markov decision process (MDP), defined by a tuple S, A, P, R, γ where S is the set of possible states of the environment; A is the set of actions available to the agent; $P(S'|s,a)$ is the transition probability function defining the likelihood of reaching state s' after taking action a in state s; $R(s,a)$ is the reward function specifying the immediate feedback for performing action a in state s; and $\gamma \in [0,1]$ is the discount factor that determines the weight of future rewards.

In many RL problems, the agent's interaction with the environment is divided into episodes, finite sequences that start from an initial state and terminate upon reaching a specific condition. These episodes allow the agent to accumulate experience over bounded time horizons and are especially useful in environments with natural reset points, such as task completion. This episodic structure provides a practical basis for evaluating and updating RL policies.

The agent's behavior is defined by a policy $\pi(a|s)$, which assigns a probability distribution over actions for each state. The goal is to learn an optimal policy π^*, which maximizes the expected cumulative discounted reward:

$$\pi^*(a|s) = \arg\max_{\pi} \mathbb{E}_{\tau \sim P^{\pi}} \big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \big]$$

where $R(s_t, a_t)$ is the immediate reward for taking action a_t in state s_t and γ^t is the discount factor that reduces the value of future rewards, thereby encouraging the agent to prefer immediate rewards. A trajectory $\tau = (s_0, a_0, s_1, \dots)$ denotes the sequence of states and actions that arises when the agent follows policy π in the environment. $\mathbb{E}[\cdot]$ represents the expectation over such trajectories, and $\arg\max_{\pi}$ identifies the policy that maximizes the expected accumulated rewards collected in the episode.

III. Related Work

Most AD research on satellites falls into one of three broad categories: research on rule-based systems, static data-driven models, or deep learning-based approaches. Rule-based techniques, such as fault detection, isolation, and recovery (FDIR), rely on predefined limits and expert-defined logic [32]. Static data-driven models apply algorithms like clustering [33], [34], SVM [9], or isolation forest [35] to detect deviations from expected behavior, typically without the need for labeled

samples. Deep learning approaches leverage temporal and multivariate patterns in telemetry, using architectures such as LSTMs [7] and AEs [8]. We focus on the third category, deep learning approaches, since rule-based and static methods depend heavily on expert knowledge and struggle when faced with unseen or evolving anomalies [36].

Deep learning methods have gained traction in satellite AD, because satellite data exhibits strong temporal and complex multivariate dependencies [1]. For example, Hundman et al. [7] demonstrated that LSTM and RNN models can learn the satellite's benign behavior and use this knowledge to detect anomalies as points where the predicted behavior deviates substantially from actual readings, identified by large reconstruction errors. Similarly, Akbarian et al. [8] proposed an LSTM-based network that utilizes an AE to extract the most important data features, which the LSTM uses to capture long-term dependencies. When focusing on CubeSat, Horne et al. [10] proposed two deep learning-based AD models (CNN and LSTM) designed specifically to run efficiently on a microcontroller implemented on a CubeSat. However, these approaches rely on a single fixed model, which cannot adapt to variations in data distributions or onboard resource constraints. Our results demonstrate that such static AD models either achieve high detection performance at the cost of excessive energy consumption or remain energy-efficient but less accurate. This adaptability allows LighTellite to sustain high detection reliability while operating near the lower energy bound.

IV. Methodology

The proposed framework's pipeline is presented in Figure 1. LighTellite aims to balance AD performance and energy efficiency by dynamically adapting to onboard data patterns and available onboard resources. The framework consists of three main phases:

1) **Phase 1:** While in orbit, LighTellite reads recent satellite data, capturing both sensor readings and short-term statistical summaries.

2) **Phase 2:** A fixed-duration segment of the data, referred to as a section, is provided to the Manager agent. Based on this section, the Manager constructs its state and determines an appropriate energy budget, a risk factor (probability for abnormal activity), and likelihoods of each attack scenario. The section and Manager outputs are then passed to the Worker agent for processing.

3) **Phase 3:** The Worker divides the section into fixed-size chunks. Using the Manager's outputs and current chunk statistics, the Worker constructs a contextual state for each chunk that captures short-term fluctuations in voltage and current, energy usage trends, and recent anomaly history. Based on this state, the Worker selects the most suitable AD model from a predefined pool and applies the model to determine whether the chunk is anomalous or not. Since each anomaly detector in the pool has a different performance and energy profile, LighTellite enables the dynamic selection of the optimal detector in response to the satellite's changing operational and energy constraints.

A. Phase 1 - Section Preparation

During each orbit, LighTellite periodically processes telemetry and system measurements such as voltage, current, and CPU usage (additional details under Experimental Settings). Each orbit O_j is modeled as a finite sequence of equal-duration segments called sections, each representing an activation period in the framework:

$$O_j = [S_{j,1}, S_{j,2}, \ldots, S_{j,n}],$$

where every section $S_{j,i}$ contains the data collected onboard during its activation period. Let $\mathbb{F} = \{F_1, F_2, \ldots, F_K\}$ denote the set of feature groups (e.g., telemetry, system) in each section, where each F_k corresponds to a subset of sensors monitoring specific aspects of onboard resource use, such as electrical usage and network activity. In each activation period, the raw data of the most recent section, together with statistical summary derived from several preceding sections, are provided to the Manager as inputs in the next phase.

B. Phase 2 - Budget and Risk Assessment

The Manager agent determines how much energy can be dedicated to AD for the current section. It receives the following inputs: the most recent section's raw data $S_{j,i}$ and a short-term statistical summary (e.g., averages and standard deviations of key telemetry features such as voltage or temperature) of several preceding sections. These summaries provide statistical context, giving the Manager a clear view of both the current operational conditions and recent trends. Using these inputs, the Manager constructs a state vector that captures the satellite's operational context, including normalized battery level, energy trends, orbit progress, section-level telemetry and system statistics, and recent anomaly history. During training, the Manager learns a policy that balances detection performance with energy efficiency. The section-level reward function guiding this optimization is constructed from several components: The detection term, active only when attacks are present, is defined as:

$$R^{det} = F + P$$

where F denotes the F1 score achieved by the Worker, and P quantifies the Worker's progress, measured as the fraction of chunk-level steps completed relative to the total expected steps for the section. The energy consumption is penalized according to the total inference energy E consumed by the Worker in the section. To discourage assignment of unnecessarily large budgets, we introduce B^{ex}, which quantifies the gap between the assigned budget and the actual energy consumption in each section. Manager's early episode termination due to battery depletion is penalized through D, which denotes the proportion of sections still remaining in the orbit at the time of termination. Finally, combining these components, the reward function for section i defined as:

$$R_i = w_1 Y_i R_i^{det} - w_2 E_i - w_3 B_i^{ex} - (1 - Y_i) w_4 L_i - w_5 D_i$$

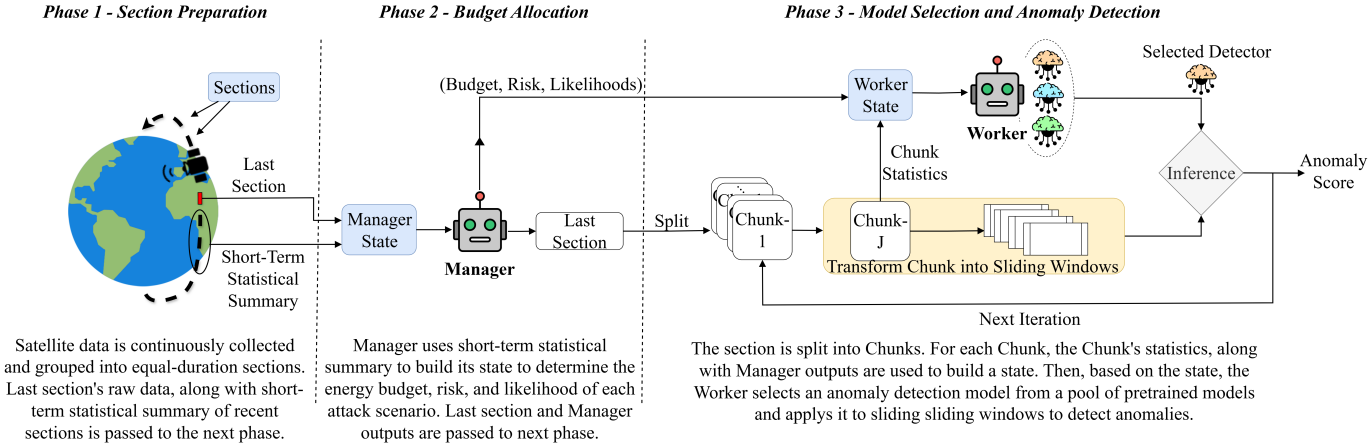

Fig. 1. The proposed LighTellite framework's pipeline.

where $Y_i \in \{0, 1\}$ indicates whether attacks are present in section i, and L_i is a penalty that increases when the Manager assigns unnecessarily high attack scenario likelihoods in benign sections.

The outputs of this phase include the assigned budget B_i, the risk factor and a likelihood for each attack scenario, and the section's raw data $S_{j,i}$, which are passed to the next phase.

C. Phase 3 - Model Selection and Anomaly Detection

The Worker agent operates within the energy budget B_i assigned by the Manager for the current section $S_{j,i}$. Its goal is to detect anomalous behavior in the section while adhering to the energy budget. To achieve this, the Worker performs DMS in a process that balances AD performance and computational costs through a sequence of localized decisions. There are three main stages in this process:

1) Data Preparation: Upon receiving the section $S_{j,i}$, the Worker divides the raw data into m smaller chunks,

$$S_{j,i} = [C_1, C_2, \ldots, C_m],$$

each covering a short, fixed-duration interval of the data. This division increases the temporal resolution of the analysis, enabling the Worker to dynamically adapt its detection strategy as operational conditions evolve. Each chunk C_t is then transformed into a sequence of sliding windows:

$$C_t = [W_{t,1}, W_{t,2}, \ldots, W_{t,y}],$$

which serve as inputs to the AD models.

2) Agent State Construction: In this stage, a contextual state s_t is constructed for each chunk. The Worker computes relevant data statistics and trends of voltage, current, and CPU usage, and combines them with the Manager's outputs. The resulting state representation provides a compact summary of the system's operational conditions and resource availability, guiding the subsequent model selection stage.

3) Model Selection and Inference: In this stage, the Worker selects and applies the most suitable AD model for each chunk based on its current state s_t. For each feature group $f \in \mathbb{F}$, two pretrained detectors are available: a lightweight model L_f with lower energy consumption and a heavier model H_f that offers higher detection accuracy at a higher energy cost. Let the candidate action set be $\mathcal{D} = \{L_f, H_f \mid f \in \mathbb{F}\}$. Given s_t, the Worker chooses the detector whose energy-performance trade-off best matches the current operational conditions and applies it to all sliding windows in the chunk to produce anomaly scores. After inference, the measured energy $E_{\text{used},t}$ is subtracted from the section's budget:

$$B_i^{(t+1)} = B_i^{(t)} - E_{\text{used},t}.$$

If the remaining budget $B_i^{(t+1)}$ falls below a predefined safety threshold, the Worker terminates further processing of the section to maintain operational safety.

4) Reward Optimization: During training, the Worker is guided by a reward function that balances model relevance and energy efficiency. The reward function encourages the selection of detectors whose features align with indicative signals of the current attack scenario; the function also favors heavier models when attacks are present and discourages their use during benign activity. Additional penalties are applied if the total energy used exceeds the assigned budget. The per-chunk reward is defined as:

$$r_t \;=\; \theta_A \,\text{Align}_t \;+\; \theta_H \, Y_t \, H_t \;-\; \lambda_E \, E_t \;-\; \lambda_B \, \mathbb{I}\big[E_t > B_i\big],$$

where Align_t indicates whether the selected detector's feature group matches the signals most relevant to the observed behavior in chunk C_t; H_t denotes that a heavy model was selected; Y_t indicates the presence of attack activity; and E_t and B_i represent the consumed and allocated energy, respectively. The coefficients θ_A, θ_H, λ_E, and λ_B control the trade-off between feature alignment, model capacity, energy efficiency, and budget compliance.

V. Evaluation

To evaluate LighTellite's effectiveness, we performed a comprehensive set of experiments using data collected from 58 full-orbit simulations on the AegisSat, a SOTA CubeSat testbed [12]. The evaluation was performed in two stages: first, we compared and profiled individual AD models trained on different feature groups (telemetry, system, and combined (both groups)) to assess how feature selection influences detection performance, model complexity, and energy efficiency; then we evaluated the complete RL framework to analyze its ability to balance energy consumption and detection performance. Each experiment described below was designed to address one or more of the following research questions.

A. Research Questions

RQ1. How does the detection performance of feature-specific anomaly detectors (trained on telemetry or system data) compare to that of a detector trained on the combined feature space across diverse attack scenarios?
RQ2. How do AD models differ in computational and energy efficiency, as reflected by their complexity (parameters and floating point operations (FLOPs)) and energy measurements?
RQ3. How does the LighTellite framework perform compared to static AD models (in which the same model is used throughout the entire orbit) in terms of energy consumption and detection performance?

B. Data Collection

To create a dataset of CubeSat simulations, we used the AegisSat testbed [12], which integrates both hardware and software components, enabling realistic modeling of orbital dynamics and onboard computing, while allowing for injection and monitoring of cyberattack scenarios. We collected synchronized telemetry and system data from 58 complete CubeSat orbit simulations, each lasting approximately 90 minutes and comprising about 5,000 samples:
1) Telemetry: measurements from the onboard electrical power system (EPS), including line voltage and current, battery voltage and current, and battery temperature.
2) System: operational data from an onboard Raspberry Pi Zero 2W, capturing CPU utilization, system activity (context switches and interrupts), and network I/O statistics (bytes transmitted and received).
Data from these sources was used to construct the training, validation, and test datasets. Note that for each of the 58 complete orbit simulations, about 5,000 samples were collected, resulting in over 230K samples, all of which are included in our open-source dataset. Of the collected 58 orbits, 23 represented nominal CubeSat operation while the other 35 (approximately 4.5% of samples out of the total dataset) included cyberattack scenarios involving excessive resource usage, data leakage, or both. An additional metadata file with more details is supplied with the open-source dataset.

C. Experimental Settings

1) Computational Environment: A workstation equipped with an NVIDIA RTX 4090 GPU, 16 GB RAM, and Python 3.11.9 was used for training, while energy measurements and evaluations were obtained on a Raspberry Pi Zero 2W implemented on the CubeSat's onboard hardware.

2) Attack Scenarios: Two attack scenarios were simulated to represent cyber threats to CubeSats. Each attack was activated in randomly selected time intervals within randomly selected orbits, and its activation periods were labeled for the performance evaluation.

Excessive Resource Usage. This scenario represents a composite stress condition combining several resource-intensive behaviors. During activation, multiple processes were launched in parallel to overload the onboard computer, including repeated file creation and deletion, continuous memory allocation, and CPU-intensive loops. Together, these activities simulate a range of malware-like behaviors that waste available onboard computing resources.

Data Leakage via Radio Frequency Communication. This scenario emulates the unauthorized transmission of onboard data to an external third-party adversary receiver. When active, the malicious process transmits locally saved images from the satellite's storage through the CubeSat's RF transmitter, reproducing a data exfiltration event that exploits legitimate communication hardware to leak sensitive information.

3) Anomaly Detection Models: All AD models were trained in an unsupervised manner, using only benign orbits, and evaluated on a labeled test set containing both benign and attacked samples. Each model is a variational autoencoder (VAE) designed to learn the normal temporal behavior of telemetry or system signals and detect deviations. To learn the trade-off between detection accuracy and energy efficiency, we varied the model capacity by adjusting the number of hidden units, latent dimensions, and temporal layers, creating lightweight and heavy configurations for each feature group (telemetry, system, and combined).

The models were trained on batches of 64 sliding windows, each containing 60 consecutive samples, created using a stride of 5 for telemetry and 12 for system. A window was labeled anomalous if at least 50% of its samples were labeled as attacks. We used the AdamW optimizer with a learning rate of 3×10^{-4}. The objective function combined a reconstruction term with a Kullback-Leibler (KL) divergence regularization on the latent space. Training was performed for up to 100 epochs, with early stopping determined on the validation set. After training, the models calculated a threshold value based on the validation set, and any score higher than the threshold value is considered anomalous. Thresholds can be determined through unsupervised criteria (e.g., quantile-based rules) or by tuning on a small labeled validation subset, with our implementation following the latter approach [37].

4) RL Agents: Each orbit was split to 8 sections, where each section divided to 8 chunks.

Worker Agent. The Worker agent was trained using the proximal policy optimization (PPO) algorithm with a discrete action space corresponding to the selection between lightweight (L_i) and heavy (H_i) AD model for each feature group f_i. Its state comprised contextual features describing the remaining energy budget, recent telemetry dynamics, system operating conditions, and short-term anomaly patterns, along with recent actions and performance trends. Training was conducted for 1K timesteps by simulating possible Manager outputs (without the actual involvement of the Manager agent) with the following PPO parameters: $n\_steps = 256$, $batch\_size = 256$, and $net\_arch = [64, 64]$.

Manager Agent. The Manager agent was trained using the PPO algorithm with a continuous four-dimensional action space controlling the section's energy budget, likelihoods for each of the two attack scenarios, and a risk factor. The Manager's state summarizes the current energy context, orbit progression (e.g., time until recharge), telemetry trends, and recent anomaly activity. Training was performed for 10K timestemps with the following PPO parameters: $n\_steps = 128$, $batch\_size = 128$, and $net\_arch = [64, 64]$. During training, the Manager interacted with the frozen pretrained Worker agent. Finally, to fine-tune the agents' coordination, we performed ten additional joint training iterations, each consisting of 40K and 5K timesteps for the Worker and Manager respectively.

5) Energy Measurement Procedure: To accurately estimate the energy consumption of the models, we employed a dedicated measurement setup. An external electrical circuit was connected to a Raspberry Pi Zero 2W, replicating the CubeSat's onboard hardware, to enable high-resolution measurement of voltage and current during model execution. A Python script ran each trained model on a fixed sequence of input samples (identical across models), while the circuit continuously recorded the energy consumption throughout the inference periods. This procedure was repeated 100 times per model to obtain a robust distribution of inference energy consumptions, from which the mean and standard deviation were computed. We exclude the energy cost of the RL agents, as their policies (two fully connected layers of 64 neurons) contributed less than 1% of the total consumption and therefore negligible relative to the AD models.

D. Results

Figures 2-5 present the performance of each individual AD model and the proposed framework, LighTellite, under a realistic hybrid attack scenario (i.e., results were not presented for each scenario separately) on the test set. Since the combined models performed worse than the domain-specific models (see Tables I), we excluded them from further analysis. Each AD model corresponds to a static configuration in which the same model is used throughout the entire orbit, whereas LighTellite dynamically selects between models during operation, based on the learned policies of the Manager and Worker agents. In the tables, we marked the best obtained results in bold.

TABLE I
DETECTION PERFORMANCE OF STATIC ANOMALY DETECTORS

Attack Scenario	Features	Model	Precision	Recall	F1
Resource Usage	Telemetry	Lightweight	0.88	0.81	0.83
		Heavy	0.90	0.91	**0.91**
	System	Lightweight	0.75	1.00	0.85
		Heavy	0.76	1.00	0.86
	Combined	Lightweight	0.91	0.71	0.79
		Heavy	0.83	0.78	0.80
Data Leakage	Telemetry	Lightweight	0.22	0.06	0.08
		Heavy	0.25	0.07	0.10
	System	Lightweight	0.72	0.99	0.83
		Heavy	0.95	0.99	**0.97**
	Combined	Lightweight	0.82	0.13	0.21
		Heavy	0.59	0.75	0.64
Hybrid	Telemetry	Lightweight	0.71	0.59	0.63
		Heavy	0.79	0.67	0.71
	System	Lightweight	0.36	1.00	0.52
		Heavy	0.37	1.00	0.53
	Combined	Lightweight	0.89	0.58	0.69
		Heavy	0.74	0.76	**0.75**

*1) **Impact of Feature Specialization (RQ1)**:* Table I presents the detection results of static AD models, each trained on one of the three feature groups: telemetry, system, and combined (combination of both feature groups). Each feature group includes two model configurations, lightweight and heavy, whose performance under three representative attack scenarios is reported: excessive resource usage, data leakage via RF communication, and hybrid. The precision, recall, and F1 score served as the evaluation metrics, where the F1 score reflects the harmonic mean between precision and recall.

Several clear patterns emerge from the results. Models trained on features that capture the primary effects of each attack achieve the best performance, e.g,. models which were trained on network (system) related features would detect data leakage attacks better. The heavy telemetry model obtained the highest F1 score (0.91) in the resource usage scenario, where electrical and power signals are most indicative of the attack's impact , while the heavy system model performed best (0.97) in the data leakage scenario, which primarily affects network and CPU activity. Telemetry-based models did not generalize to data leakage scenario, and both telemetry and system models showed reduced performance on the hybrid scenario, reflecting the limited cross-domain transferability of domain-specific knowledge. As expected, across the three feature groups, the lightweight models performed slightly worse (in terms of the F1 score) than the heavy models. Models trained on the combined feature group achieved moderate performance in the hybrid attack scenario, where both feature types contribute, but underperformed compared to domain-specific detectors in their respective domains.

Figure 2 presents the true positive rate (TPR) achieved after training the complete LighTellite framework. TPR represents the proportion of attacked sliding windows correctly detected,

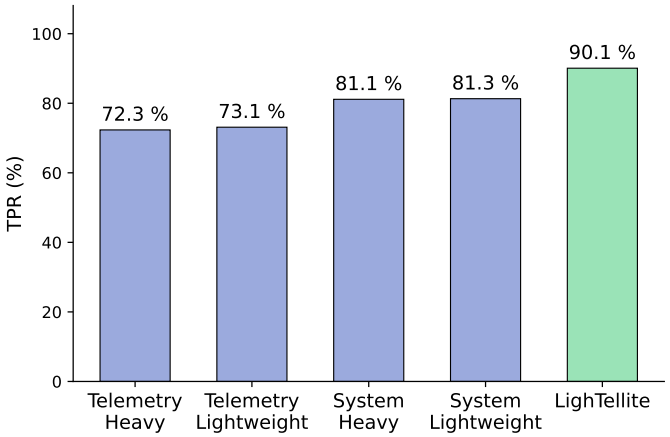

Fig. 2. Comparison of TPR, representing the detection rate of cyber attacks, achieved by each static AD model and the proposed LighTellite framework under a realistic hybrid attack scenario. LighTellite achieves the highest TPR (90.1%), substantially outperforming all static AD models (72.3-81.3%).

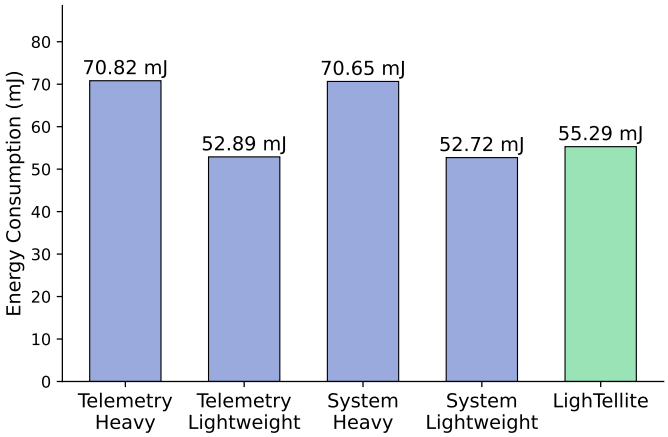

Fig. 3. Comparison of the average energy consumed, (measured in millijoules) across an entire orbit. The heavy AD models require approximately 71 mJ per orbit, while lightweight models consume around 53 mJ. LighTellite operates at 55.29 mJ on average, about 4.7% higher than the lightweight AD models yet up to 21.8% lower than the heavy variants.

providing a direct measure of detection effectiveness. LighTellite outperforms all static AD models, obtaining a TPR of 90.1% compared to 72.3-81.3% obtained by the static models.

TABLE II
MODEL CAPACITY AND ENERGY DISTRIBUTIONS OF STATIC ADS

Features	Model	Parameters	MFLOPS	Energy Distribution (mJ)	
				Mean	Std.
Telemetry	Lightweight	14,519	1.40	6.61	0.07
	Heavy	37,715	3.77	8.85	0.06
System	Lightweight	14,348	1.38	6.59	0.07
	Heavy	37,448	3.73	8.83	0.06

*2) **Model Complexity and Energy Efficiency (RQ2):*** Table II summarizes the capacity and inference energy consumption of static AD models. For each pair of AD models (lightweight and heavy), that were trained on different feature group, we report the number of learnable parameters, the approximate number of floating-point operations during inference (in millions, MFLOPs), and the measured energy distribution in millijoules. These energy distributions are later used in the LighTellite framework to sample the energy consumption of each model during inference.

As expected, model complexity strongly correlates with inference cost. Heavy models contain roughly 3x more parameters and MFLOPs than their lightweight counterparts and consume about 8.8 mJ per inference, compared to 6.6 mJ for the lightweight variants. This consistent gap highlights the clear trade-off between computational capacity and energy efficiency. Telemetry and system models show nearly identical energy patterns, confirming that model architecture, rather than input dimensionality, influences energy consumption.

*3) **LighTellite's Performance and Dynamic Adaptation (RQ3):*** As seen in Figures 2 and 3, the heavy models consume approximately 71 mJ per orbit, while the lightweight models require about 53 mJ. LighTellite operates at 55.29 mJ on average, about 4.7% higher than the lightweight AD models and up to 21.8% lower than the heavy variants. The lightweight models thus represent the theoretical lower bound of energy consumption, whereas LighTellite's slightly higher value stems from its adaptive behavior; during operation, it dynamically selects models based on the learned policies of the Manager and Worker agents. The small difference in energy consumption indicates that LighTellite mainly employs lightweight models, using heavier ones only when necessary.

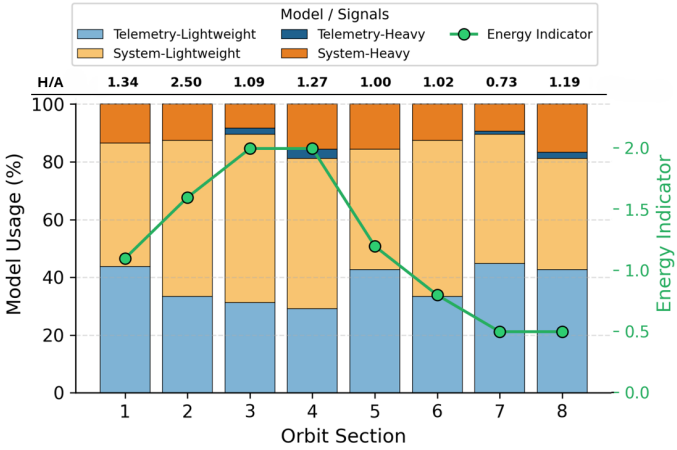

Fig. 4. Activation frequency of each AD model selected by the Worker agent across orbit sections (stacked bars, left Y-axis). The value above each bar shows the H/A ratio, defined as the proportion of heavy models usages divided by the proportion of anomalous samples in that section. The green line represents the Energy Indicator (right Y-axis), reflecting the CubeSat's energy conditions.

Figure 4 illustrates the dynamic model selection (DMS) pattern learned by the Worker agent. Each bar represents the usage proportion (Y-axis on the left) of each model (selected by the Worker agent for inference) across the satellite sections (X-axis), showing how LighTellite alternates between lightweight and heavy detectors. Lightweight models dominate

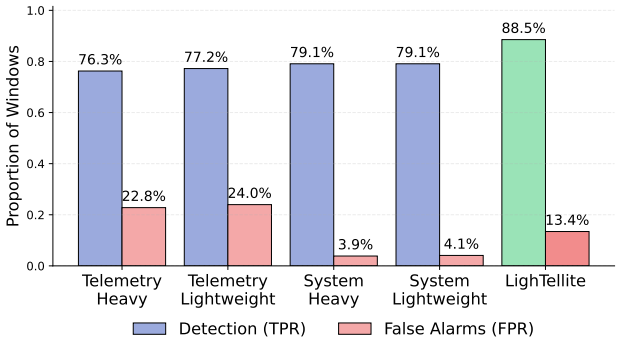

Fig. 5. Comparison of the TPR (proportion of attacks detected) and FPR (proportion of benign windows incorrectly flagged as attack) for each static AD model and the LighTellite framework.

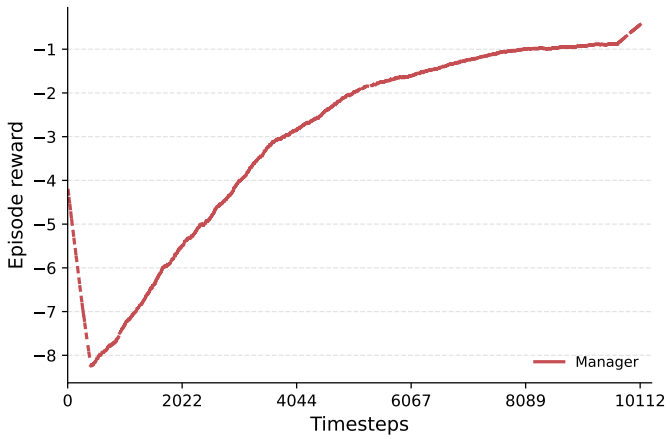

Fig. 6. Curve showing how the episode rewards evolve during training, with each point representing the smoothed mean episode reward of the Manager agent over training timesteps.

most sections (brighter colors), reflecting the framework's preference for energy-efficient inference during normal operation, while heavy models are employed more selectively when anomalous activity is likely to occur. H/A ratio in each section (averaged across all orbits) is shown above each bar:

$$H/A = \frac{\text{Proportion of heavy models usages}}{\text{Proportion of anomalous samples}}$$

where values greater than 1 indicate that heavy models were used more frequently than anomalous activity occurred in that section. In addition, the green line across the bars shows the Energy Indicator, which reflects the CubeSat's energy conditions, with its values displayed on the right Y-axis. When the energy conditions are favorable (Sections 1-4, where the green line increases), the H/A ratio is higher (1.09-2.50). In contrast, when the energy conditions are constrained (Sections 5-8), the system reduces its use of heavy models, even when anomalies are present, to avoid unnecessary energy consumption, resulting in lower H/A values (0.73-1.19). In Section 8, the Energy Indicator increases again, enabling LighTellite to select heavy models, raising the H/A ratio from 0.73 to 1.19.

To analyze the AD behavior of the models, we first computed a confusion matrix aggregated across all test orbits. Figure 5 presents the TPR and FPR illustrating how reliably each method detects attacks and how often it raises false alarms. LighTellite achieves the highest TPR, confirming its ability to outperform static AD models in attacks detection while maintaining FPR relatively low.

*4) **Policy Optimization**:* To evaluate how effectively the RL components learned their decision policies, we analyzed the reward progression of the Manager agent during training (as shown in Figure 6). The plotted curve shows the smoothed mean episode reward over training timesteps. Because it represents the Manager's reward, which incorporates the Worker's detection performance and energy usage, the curve effectively captures the behavior of both agents. This provides a clear view of how the framework's decision-making improved throughout training.

VI. Discussion

A. Limitations

External Operating Factors. Although LighTellite obtained strong results, external operating factors, such as radiation exposure, temperature fluctuations, and gradual hardware aging, may introduce noise that affects sensor readings and model stability, potentially reducing performance. To mitigate this, periodic in-orbit recalibration using onboard data can address these effects, ensuring that reliable detection is maintained over time.

Dependence on Model Pool Diversity. The framework's performance depends on the quality of the pretrained AD models in its model pool. Utilizing more advanced architectures trained on up-to-date data distributions can enhance robustness and ensure that the framework remains aligned with the satellite's evolving data characteristics.

Transition to Real Missions. Although the AegisSat testbed provides a realistic environment for evaluation, recorded simulations cannot fully capture all in-orbit conditions such as long-term hardware degradation and radiation effects. This limitation can be addressed by using data recorded from real orbiting CubeSats and incorporating additional onboard parameters, ensuring that the learned policies generalize effectively to real mission environments.

B. Broader Implications

Energy-Aware Framework for AD. This work contributes to the movement toward sustainable AI by showing that energy awareness can serve as a core design principle for AD systems in space. Rather than treating efficiency as a secondary goal, LighTellite demonstrates how energy awareness can be integrated directly into the learning and decision process. This understanding could prompt the use of additional performance metrics and development of new AD frameworks that consider both accuracy and resource utilization.

RL-Based Dynamic Model Selection for Space Systems. This study demonstrates that RL can enable dynamic, context-

aware decision-making in autonomous spacecraft operating under resource constraints. Beyond improving AD, it establishes RL as a general control mechanism for adaptation to changing mission conditions and energy availability.

Generalization Beyond the CubeSat Domain. Although LighTellite was designed for small satellites, its principles can extend to other domains characterized by limited energy and computational capacity. The framework's modular, energy-aware, and adaptive nature makes it relevant to areas such as IoT networks and autonomous vehicles. Thus, this work provides a general foundation for building intelligent systems that maintain performance while adhering operational constraints in diverse environments.

VII. Conclusion and Feature Work

We introduced LighTellite, a RL-based framework that dynamically selects suitable AD models, using cooperative Manager-Worker agents. By embedding energy awareness directly into the decision process, LighTellite provides a principled approach for achieving sustainable and adaptive intelligence on resource-constrained platforms.

Experimental results show that LighTellite optimizes the trade-off between energy consumption and performance better than static AD models. The results highlight the impact of using domain-specific cues to guide DMS and adaptive decision-making, demonstrating that contextual awareness is key to achieving efficiency without sacrificing performance.

Future work might include introducing a wider range of attack scenarios, enabling evaluation of the framework's robustness under varied operational conditions. In addition, expanding the static anomaly detectors pool with novel detectors and additional feature sets could improve adaptability and overall detection coverage.

References

[1] A. Fejjari, A. Delavault, R. Camilleri, and G. Valentino, "A review of anomaly detection in spacecraft telemetry data," *Applied Sciences*, vol. 15, no. 10, p. 5653, 2025.

[2] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[3] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM computing surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[4] T. Villela, C. A. Costa, A. M. Brandão, F. T. Bueno, and R. Leonardi, "Towards the thousandth cubesat: A statistical overview," *International Journal of Aerospace Engineering*, vol. 2019, no. 1, p. 5063145, 2019.

[5] N. Saeed, A. Elzanaty, H. Almorad, H. Dahrouj, T. Y. Al-Naffouri, and M.-S. Alouini, "Cubesat communications: Recent advances and future challenges," 2020. [Online]. Available: https://arxiv.org/abs/1908.09501

[6] I. V. McLoughlin, V. Gupta, G. Sandhu, S. Lim, and T. Bretschneider, "Fault tolerance through redundant cots components for satellite processing applications," in *Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint*, vol. 1. IEEE, 2003, pp. 296–299.

[7] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 387–395.

[8] H. Akbarian, "Deep learning based anomaly detection in space systems and operations," Ph.D. dissertation, Florida Atlantic University, 2024.

[9] Z. Xu, Z. Cheng, and B. Guo, "A hybrid data-driven framework for satellite telemetry data anomaly detection," *Acta Astronautica*, vol. 205, pp. 281–294, 2023.

[10] R. Horne, S. Mauw, A. Mizera, A. Stemper, and J. Thoemel, "Anomaly detection using deep learning respecting the resources on board a cubesat," *Journal of Aerospace Information Systems*, vol. 20, no. 12, pp. 859–872, 2023.

[11] V. Mehlin, S. Schacht, and C. Lanquillon, "Towards energy-efficient deep learning: An overview of energy-efficient approaches along the deep learning lifecycle," *arXiv preprint arXiv:2303.01980*, 2023.

[12] R. Idan, R. Peled, A. B. S. Tov, E. Markus, B. Zadov, O. Chodeda, Y. Fadida, O. Holschke, J. Plachy, A. Shabtai *et al.*, "Aegissat: A satellite cybersecurity testbed," 2025.

[13] C. Feng and J. Zhang, "Reinforcement learning based dynamic model selection for short-term load forecasting," in *2019 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 2019, pp. 1–5.

[14] Y. Fu, D. Wu, and B. Boulet, "Reinforcement learning based dynamic model combination for time series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 6, 2022, pp. 6639–6647.

[15] J. E. Zhang, D. Wu, and B. Boulet, "Time series anomaly detection via reinforcement learning-based model selection," in *2022 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2022, pp. 193–199.

[16] Y. Wang, L. Xiong, M. Zhang, H. Xue, Q. Chen, Y. Yang, Y. Tong, C. Huang, and B. Xu, "Heat-rl: online model selection for streaming time-series anomaly detection," in *Conference on Lifelong Learning Agents*. PMLR, 2022, pp. 767–777.

[17] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 409–424.

[18] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8817–8826.

[19] A. Jamalipour and T. Tung, "The role of satellites in global it: trends and implications," *IEEE personal communications*, vol. 8, no. 3, pp. 5–11, 2001.

[20] J. Yang, P. Gong, R. Fu, M. Zhang, J. Chen, S. Liang, B. Xu, J. Shi, and R. Dickinson, "The role of satellite remote sensing in climate change studies," *Nature climate change*, vol. 3, no. 10, pp. 875–883, 2013.

[21] O. Ledesma, P. Lamo, and J. A. Fraire, "Trends in lpwan technologies for leo satellite constellations in the newspace context," *Electronics*, vol. 13, no. 3, p. 579, 2024.

[22] R. Gubby and J. Evans, "Space environment effects and satellite design," *Journal of atmospheric and solar-terrestrial physics*, vol. 64, no. 16, pp. 1723–1733, 2002.

[23] J. Howard, *Spacecraft environments interactions: Space radiation and its effects on electronic systems*. NASA, 1999.

[24] U. Shakoor, M. Alayedi, and E. E. Elsayed, "Comprehensive analysis of cubesat electrical power systems for efficient energy management," *Discover Energy*, vol. 5, no. 1, pp. 1–20, 2025.

[25] H. Heidt, J. Puig-Suari, A. Moore, S. Nakasuka, and R. Twiggs, "Cubesat: A new generation of picosatellite for education and industry low-cost space experimentation," 2000.

[26] S. S. Arnold, R. Nuzzaci, and A. Gordon-Ross, "Energy budgeting for cubesats with an integrated fpga," in *2012 IEEE Aerospace Conference*. IEEE, 2012, pp. 1–14.

[27] R. Peled, E. Aizikovich, E. Habler, Y. Elovici, and A. Shabtai, "Evaluating the security of satellite systems," *arXiv preprint arXiv:2312.01330*, 2023.

[28] G. Marra, U. Planta, P. Wüstenberg, and A. Abbasi, "On the feasibility of cubesats application sandboxing for space missions," *arXiv preprint arXiv:2404.04127*, 2024.

[29] V. Knap, L. K. Vestergaard, and D.-I. Stroe, "A review of battery technology in cubesats and small satellite solutions," *Energies*, vol. 13, no. 16, p. 4097, 2020.

[30] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[31] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[32] F. SalarKaleji and A. Dayyani, "A survey on fault detection, isolation and recovery (fdir) module in satellite onboard software," in *2013 6th international conference on recent advances in space technologies (rast)*. IEEE, 2013, pp. 545–548.

[33] M. A. Obied, F. F. Ghaleb, A. E. Hassanien, A. M. Abdelfattah, and W. Zakaria, "Deep clustering-based anomaly detection and health monitoring for satellite telemetry," *Big Data and Cognitive Computing*, vol. 7, no. 1, p. 39, 2023.

[34] Y. Gao, T. Yang, M. Xu, and N. Xing, "An unsupervised anomaly detection approach for spacecraft based on normal behavior clustering," in *2012 Fifth International Conference on Intelligent Computation Technology and Automation*. IEEE, 2012, pp. 478–481.

[35] J. Murphy, J. E. Ward, and B. Mac Namee, "An overview of machine learning techniques for onboard anomaly detection in satellite telemetry," in *2023 European Data Handling & Data Processing Conference (EDHPC)*. IEEE, 2023, pp. 1–6.

[36] M. Kohli and I. Chhabra, "A comprehensive survey on techniques, challenges, evaluation metrics and applications of deep learning models for anomaly detection," *Discover Applied Sciences*, vol. 7, no. 7, p. 784, 2025.

[37] A. Komadina, M. Martinić, S. Groš, and Ž. Mihajlović, "Comparing threshold selection methods for network anomaly detection," *IEEE access*, 2024.