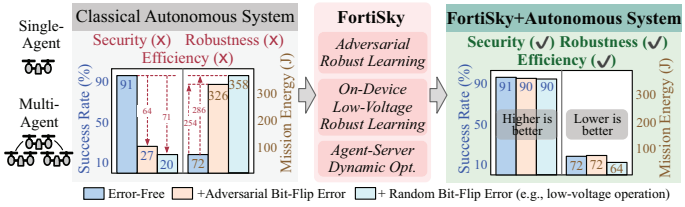
FortiSky: Enhancing Adversarial and Bit-Error Robustness for Efficient and Secure Autonomous Systems

Zishen Wan[1†], Karthik Swaminathan[2], Nandhini Chandramoorthy[2], Pin-Yu Chen[2],
Tushar Krishna[1], Vijay Janapa Reddi[3], Arijit Raychowdhury[1]
[1]*Georgia Institute of Technology* [2]*IBM Research* [3]*Harvard University*

***Abstract*—Autonomous systems, such as unmanned aerial vehicles (UAVs), are required to employ complex AI models to execute fully autonomous position-navigation-timing missions. However, deploying such functionalities on UAVs remains challenging due to stringent onboard size, weight, and power constraints. Further, these UAVs may also be vulnerable to adversarial attacks in complex real-world environments. Existing methods often address either efficiency or robustness, but rarely both, frequently neglecting or even compromising one to optimize the other.**

To this end, we propose FortiSky, a robust learning framework to jointly enhance robustness to both adversarial and random bit errors, realizing efficient and secure UAV systems. FortiSky supports both single-agent and multi-agent robust learning, both offline and on-device, with adaptive and collaborative optimizations. FortiSky is the first design that ensures both adversarial robustness and high energy efficiency enabled by very-low voltage operation onboard UAVs. Through extensive system-level UAV experiments combining algorithm-level robust learning and hardware-level silicon tests, FortiSky achieves 3.73× processing energy reduction and 14.6% mission energy reduction, thus effectively co-optimizing efficiency and robustness of onboard UAV.



	Adversarial Robustness	Bit-Error Robustness	Learning Support	Targeted System	Multi-Agent Support	Protection Technique
Dante [15]	✗	✓	✗	SL	✗	Hardware
RandBET [16]	✗	✓	✓	SL	✗	Software
DAJAT [17]	✓	✗	✓	SL	✗	Software
ComA-FedRL [18]	✓	✗	✗	RL	✓	Software
BERRY [6]	✗	✓	✓	RL	✗	Software
FortiSky (Ours)	✓	✓	✓	RL	✓	Software

Targeted System: SL - Supervised Learning; RL - Reinforcement Learning.

Fig. 1: FortiSky Enables *Robust* and *Efficient* Autonomous Systems. FortiSky is a unified robust learning framework enhancing both adversarial and low-voltage bit-error robustness to unlock secure and efficient processing for single- and multi-agent autonomous systems.

I. Introduction

The adoption of autonomous systems is becoming increasingly widespread in position-navigation-timing tasks [1]–[4]. To achieve the required level of autonomy, Unmanned Aerial Vehicles (UAVs) are expected to execute complex Reinforcement Learning (RL) models onboard, often operated in challenging and dynamic environments. However, deploying these systems onboard real-world devices presents significant challenges. *First*, the computational complexity that such PNT tasks entail, conflicts with stringent Size, Weight, and Power (SWaP) constraints inherent to UAVs [5]–[9], making it imperative to boost mission energy efficiency. *Second*, the autonomy AI models used in many safety-critical autonomous tasks are vulnerable to errors and adversarial attacks, making it challenging to ensure reliable performance in dynamic and unpredictable environments [10]–[12]. It is thus essential to develop techniques boosting both efficiency of autonomous systems as well as their robustness to both random and adversarial errors.

To tackle the first challenge of energy efficiency, an effective approach is to lower voltage of the onboard processor, given the quadratic relation between energy and voltage. In particular, reducing processing power enables it to be re-targeted towards increasing UAV flight speed, thus resulting in mission energy savings [13], [14]. Moreover, voltage scaling methods can be applied in conjunction with other software and hardware energy-saving methods. However, operating below rated voltage ranges can result in random memory bit errors [15], resulting in adverse implications on UAV mission success.

In addition to random bit errors introduced on account of aggressive voltage scaling, the processors onboard UAVs may also be vulnerable to adversarial attacks. For example, Rowhammer-based attacks on on-chip DRAMs storing models can drastically degrade inference accuracy by simply flipping a small number of bits across the autonomy model parameters [10]. Even in multi-UAV scenarios, such adversarial bit flips in a small subset of UAVs can go undetected, potentially corrupting the unified autonomy model shared among the fleet.

To tackle the second challenge of safety-critical mission robustness, recent works have proposed techniques to mitigate the effects of bit errors, both random and adversarial. Circuit/architecture/system co-optimizations [15], [19]–[21] and algorithmic methods such as error-aware training techniques [6], [16], [22], [23] are effective in addressing random bit errors. However, these methods are ineffective against adversarial attacks and may introduce power and area overheads, which are untenable for SWaP-constrained UAV systems.

Parallel efforts have sought to enhance robustness against adversarial attacks. Adversarial training is one of the most effective methods [17], [24], to address the robustness challenge. However, these methods focus on supervised learning and fail to address the sequential decision-making processes critical to RL-based autonomous systems. Moreover, they often lack robustness against random bit errors, limiting their applicability to low-voltage operations for energy-efficient SWaP-constrained

†Corresponding email: zishenwan@gatech.edu

978-3-9826741-1-7/DATE26/© 2026 EDAA

systems. *Notably, a model trained to be robust to random bit errors may still remain vulnerable to adversarial errors, and vice-versa*, highlighting the imperative need for holistic solutions that address both challenges concurrently.

In this paper, we aim to answer three fundamental questions: (1) How do adversarial and random bit errors impact RL-based UAV systems? (2) What is the correlation between adversarial and bit-error robustness? (3) How can we provide a unified framework that is robust to both adversarial and bit-flip errors, enabling both secure and energy-efficient autonomous systems?

To this end, we introduce FortiSky, a unified robust learning framework to enhance both adversarial and low-voltage bit-error robustness, delivering security, resilience, and energy efficiency for single-agent and multi-agent autonomous systems (Fig. 1). FortiSky addresses the interplay between adversarial and bit-error robustness through a cross-layer approach, integrating *algorithm-level* error-aware learning, *system-level* adaptive and collaborative optimizations, and *hardware-level* silicon characterizations. FortiSky enables computationally secure and efficient systems, even under adversarial and low-voltage operation, significantly improving mission robustness and efficiency of SWaP-constrained autonomous systems.

This paper, therefore, makes the following contributions:

• We analyze the impact of adversarial errors and low-voltage processing on single-agent and multi-agent UAV systems, and empirically illustrate the bounding generalization and correlation between adversarial and bit-error robustness.

• We present FortiSky, the unified robust learning framework to enhance both adversarial and low-voltage bit-error robustness of UAV systems. FortiSky incorporates two-stage robust learning and collaborative optimization techniques for both autonomous mission security and energy efficiency.

• By evaluating FortiSky extensively on single- and multi-UAV scenarios in the presence of adversarial and random bit errors, via cross-layer application-to-silicon analysis. We show that FortiSky achieves success rates comparable with error-free missions, while consuming 3.73× less processing energy and achieving 14.6% energy savings on navigation tasks.

II. Learning-Based Autonomous Systems

In RL-based autonomous systems, the agents learn a policy by interacting with the environment to achieve the defined goals. The learning is modeled by a Markov Decision Process (MDP) as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the MDP transition probabilities, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. At each step k, the agent observes the tuple $\mathcal{D}^k = (s^k, a^k, s^{k+1}, r^k)$, where $s^k, s^{k+1} \in \mathcal{S}$ is the current and next state, $a^k \in \mathcal{A}$ is the action taken at step k, and $r^k = \mathcal{R}(s^k, a^k)$ is the reward.

Single-Agent Autonomous System. In a single-agent systems, the goal is to learn an optimal policy π^* given the observed $\mathcal{D}$ that can maximize the reward, i.e., $\pi^*(s) = \text{argmax}_a\, Q^*(s, a)$, with the function $Q^* : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The agent uses Q-learning to update the Q function:

$$Q^{\pi}(s_t, a_t) \leftarrow \left[\mathcal{R}(s_t, a_t) + \gamma \max_{a'} Q^{\pi}(s_{t+1}, a')\right] \quad (1)$$

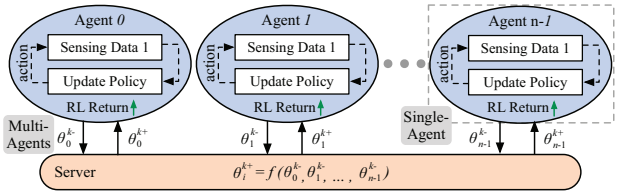


Fig. 2: Learning-Based Autonomous System. Each *single-agent* uses RL to learn autonomy model. In *multi-agent* scenarios, all agents jointly learn a unified model by interacting with environments and communicate with the server without sharing local data.

The policy converges to an optimal π^*. To minimize computational complexity, we use the Deep Q-Network (DQN) approximation f_θ: $\mathcal{S} \to \mathcal{A}$ to estimate Q parameterized by weights θ. This model learns an updated mapping from $s \to Q(s, \cdot)$ by minimizing the loss between the predicted and Bellman updated Q-values (Eq. 1). Prior works have shown that DQN performs well in UAVs [25].

Multi-Agent Autonomous System. In multi-agent systems, each agent acts and makes observations in its own environment and shares the information with a centralized server, with the aim of jointly learning a unified model π^* (Fig. 2). Each agent adopts an RL procedure as a single-agent system where the MDP at each agent i can be modeled as $\mathcal{M}_i = (\mathcal{S}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{R}_i, \gamma)$. We use θ to model the family of policies $\pi_\theta(a|s)$ and $\boldsymbol{\rho} = [\rho_0, \ldots, \rho_{n-1}]^T$ and ρ_i as the initial state distribution over $\mathcal{A}_i$ of agent i. The goal is to find θ^* that:

$$\theta^* = \arg\max_{\theta} Q(\boldsymbol{\rho}, a; \theta) \triangleq \sum_{i=0}^{n-1} \mathbb{E}_{s_i \sim \rho_i} Q_i^{\pi_\theta}(s_i, a_i) \quad (2)$$

To find the unified model in Eq. 2, $\mathcal{M}_i$ remains at the local agent i while the model θ_i is shared with the designated server. Each agent i learns its task by utilizing its local data $\mathcal{D}_i$ to train θ_i. After k steps, agents share their model θ_i^{k-} with the server. The server carries out a weighted aggregate and generates n new sets of parameters θ_i^{k+}, one for each agent, with $\theta_i^{k+} = \alpha^k \theta_i^{k-} + \beta^k \sum_{j \neq i} \theta_j^{k-}$, where $\alpha^k, \beta^k = \frac{1-\alpha^k}{n-1} \in (0, 1)$ are aggregate weights. The goal of this weighted aggregate is to achieve a consensus among agents, i.e. $\lim_{k\to\infty} \theta_i^{k+} \to \theta^*, \forall i \in \{0, n-1\}$. As the learning proceeds, the weighted aggregate constants converge to $\alpha^k, \beta^k \to \frac{1}{n}$ [26].

In this paper, we consider both single- and multi-UAV systems, which are initially trained *offline* in meta-environments. The acquired knowledge is then transferred to real-world environments and fine-tuned for *on-device* deployment.

III. FortiSky Threat Model and Impact

This section first introduces the practical fault models for adversarial errors (Sec. III-A) and low voltage-induced random bit errors (Sec. III-B). We then evaluate how these errors impact the robustness and efficiency of UAVs (Sec. III-C).

A. Adversarial Threat Model

Safety-critical UAV systems rely on autonomy models executed on onboard processors, making them vulnerable to adversarial attacks [27]. For instance, given knowledge of the data layout in memory and addressing schemes, an adversary can

Algorithm 1 Adversarial bit errors ADVBITERROR.

1: **procedure** ADVBITERRORS(θ, ϵ)
2: *// Perturb model weights by flipping at most ϵ bits*
3: Initialize $\tilde{\theta}^0$ subject to $d_H(\tilde{\theta}^0, \theta) \leq \epsilon$ and $d_H(\tilde{\theta}^0_i, \theta_i) \leq 1$
4: **for** *time step* $k = 1$ *to* T **do**
5: *// Fixed batch $\{(s_b, a_b, r_b, s_{b+1})\}_{b=1}^{B}$ from D, forward + backward pass*
6: Set $y_b = r_b + \gamma \max_{a'} Q(s_{b+1}, a'; \theta^{k-1})$
7: $\Delta^{k-1} = \nabla_\theta \sum_{b=1}^{B} \big(Q(s_b, a_b; \theta^{k-1}) - y_b\big)^2$
8: *// Adversarial errors based on gradient and project onto Hamming constraints*
9: $\tilde{\theta}^k = \tilde{\theta}^{k-1} + \alpha\, \Delta^{k-1}$
10: Project $\tilde{\theta}^k$ onto $d_H(\tilde{\theta}^k, \theta) \leq \epsilon$ and $d_H(\tilde{\theta}^k_i, \theta_i) \leq 1$
11: **end for**
12: **return** Adversarially perturbed autonomy model $\tilde{\theta}$

use Rowhammer [28] to induce bit flips in the memory arrays of UAV processors using malicious programs. In this work, we constrain the number of induced bit errors to ϵ, following the principle of L_p-constrained adversarial attacks [29], allowing at most one bit-flip per weight value to ensure the strongest adversarial attack to avoid easy detection by adversaries. In practice, not all memory bits are vulnerable to attack, however, we consider a stronger threat model from a robustness perspective. Our adversarial threat model is defined as:

Adversarial Threat Model: *An adversary, having knowledge of autonomy model and its layout in processor memory, can flip up to ϵ bits for each agent, at most one bit per weight, to degrade the autonomous system safety and quality-of-flight.*

Algo. 1 details our adversarial threat model. Following the projected gradient ascent approach [24] and letting d_H be the (bit-level) Hamming distance [30], we intend to maximize cross-entropy loss on a mini-batch $\{(s_b, a_b, r_b, s_{b+1})\}_{b=1}^{B}$ (line 5-7). We adversarially inject errors and perform a projection onto the Hamming constraints $min_{\tilde{\theta}'} ||\tilde{\theta} - \tilde{\theta}'||_2^2$, s.t. $d_H(\tilde{\theta}, \tilde{\theta}') \leq \epsilon$, $d_H(\tilde{\theta}_i, \tilde{\theta}'_i) \leq 1$, optimizing over $\tilde{\theta}'$ which will be the perturbed model after projection (line 8-10). We adopt best practices from gradient norm and momentum [30].

B. Low-Voltage Bit Errors

Reducing operating voltage is a highly effective means of improving energy efficiency due to the quadratic relationship between energy and voltage. However, reducing voltage towards near-threshold levels exacerbates bit cell variations, leading to an exponential increase in bit error rates (BERs). For our analysis, we extract the relationship between voltage, BER, and energy based on data from FinFET SRAM chips [15].

At a given voltage, these bit flips remain consistent across multiple reads and writes to the same memory location. The locations of the bit flips are random and independent across chips and arrays [15], [16], [31], [32]. To evaluate the impact of low-voltage processing, we use the following error model:

Low-Voltage Error Model: *The probability of a bit error is denoted as p. For a fixed array, errors are persistent across voltages, i.e., errors occurring at probability $p' \leq p$ also occur at p. Bit error injection is denoted as $BErr_p$.*

Notably, these voltage-induced errors are not transient and cannot be mitigated through spatial or temporal computational redundancy. Standard error-correction codes (ECC) fall short, as there can be *multiple* faulty bits within single memory word.

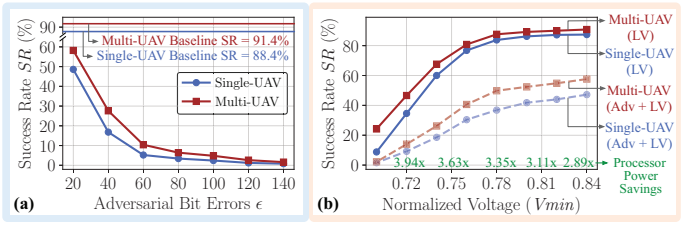


Fig. 3: Impact of Adversarial and Random Bit Errors. (a) Introduction of adversarial errors results in a sharp drop in mission success rate, which is exacerbated with increasing number of adversarial bit errors (ϵ). **(b)** Low voltage-induced memory errors degrade mission success rate more gradually. However, the addition of adversarial errors makes the system even more vulnerable to bit-flips. Note that *LV*: low-voltage error, *Adv*: adversarial attack ($\epsilon = 40$).

C. Impact of Adversarial and Low-Voltage Errors

We evaluate the effects of adversarial errors and low-voltage operations on single- and multi-agent systems, where UAVs collaboratively navigate from start to finish in the shortest time without colliding with obstacles (Sec. V). Fig. 3 shows the average mission success rate of UAVs under various scenarios.

Adversarial Error Impact. Fig. 3a demonstrates the significant degradation in UAV system safety caused by adversarial bit errors. With increasing number of adversarial errors (denoted by ϵ), agents perform more wrong actions, resulting in increased collisions with obstacles, which in turn increases overall mission time and energy. Additionally, compared with prior attacks such as Bit-Flip Attack (BFA) [33], we find ADVBITERROR delivers more effective and efficient attacks. At $\epsilon = 60$, BFA achieves a success rate of 23.5%, while the success rate in ADVBITERROR drops to $< 10\%$. BFA needs 2-3 seconds per attack iteration with the number of bit flips tied to the iterations, while ADVBITERROR requires only 0.35-0.4 seconds per iteration, and runtime is independent of ϵ.

Low-Voltage Error Impact. Fig. 3b illustrates the gradual degradation in mission success rate, caused by bit errors from low operating voltage, worsening as voltage decreases. The task success rate drops to 80% at $0.76Vmin$. Although low-voltage operation reduces onboard power, the mission detours will result in longer flight distances and extended flight time and energy. Additionally, the introduction of adversarial errors exacerbates the vulnerability to low-voltage faults, causing the success rate to decline more rapidly as voltage scales down.

Bounding Generalization to Bit Errors. To establish a probabilistic bound on the deviation between expected robust error and empirical results, we leverage Hoeffding's inequality and union bound to derive the following proposition:

$$P\left(\frac{1}{nl}\sum_{j=1}^{n}\sum_{i=1}^{l}\mathbb{1}_{Err(f(s_j,a_j;\tilde{\theta}_i))} - P(Err(f(s,a;\tilde{\theta}))) \geq \epsilon\right) \leq (n+1)e^{-n\epsilon^2\frac{l}{(\sqrt{l}+\sqrt{n})^2}}$$

where $\tilde{\theta}_i$, i=1 to l represents l instances of weights with bit errors (each bit flipped with probability p). With large l, this bound ensures that empirical test error closely approximates the expected error with the same margin across scenarios.

Single- and Multi-UAV System Comparison. Fig. 3 also highlights that multi-UAV systems exhibit greater robustness

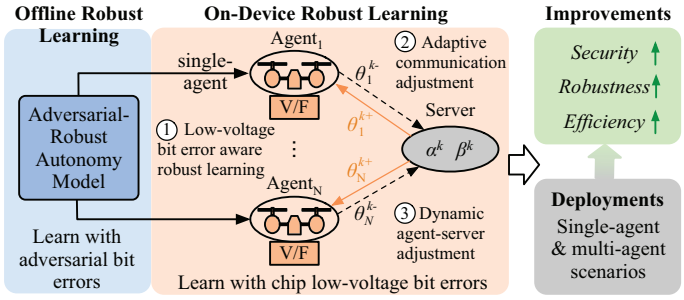


Fig. 4: FortiSky Robust Learning Framework. FortiSky supports adversarial and bit error robust learning for both single- and multi-agent systems, with offline/on-device learning, adaptive agent-server optimizations, to improve security and efficiency of UAV systems.

compared to single-UAV systems under both adversarial and low-voltage errors. Multi-UAV systems can withstand higher ϵ attacks and operate at lower voltages. This may result from the collaborative nature of multi-UAV systems, where models trained on locally collected sensor data are shared via server, enhancing overall system resilience. FortiSky aims to improve robustness and efficiency of both single- and multi-UAV systems through the unified robust learning framework.

IV. FortiSky Robust Learning Framework

This section provides an overview of the FortiSky framework and describes how FortiSky supports both adversarial and low-voltage robust operations with a host of optimizations to enhance the security and efficiency of UAV systems.

Overview. FortiSky is a robust learning framework to enhance both adversarial and bit error robustness in autonomous systems (Fig. 4). FortiSky is built upon MulBERRY [23] and enables very low-voltage operation of UAVs by developing RL-based autonomy models robust to *random bit errors* in their weights. This also improves security against manipulation of voltage settings [34]. Besides, we also address UAV robustness against a limited number of *adversarial bit errors*. In general, UAV robustness to bit errors is a desirable goal to maintain safe operation and facilitate low-power autonomous system design.

FortiSky begins with a round of *offline adversarial robust learning* to equip UAV agent(s) with adversarial robust models. These models are then deployed onboard UAVs for real-world missions. To enable robust low-voltage operations and achieve significant energy savings, FortiSky further conducts *on-device low-voltage robust learning*, where agents learn on low-voltage devices subject to bit errors and subsequently execute navigation tasks on the same hardware. In multi-UAV scenarios, each UAV learns using its own data on the low-voltage device incurring errors, and then performs the navigation mission on the same hardware. Each UAV communicates with the server at certain periodic intervals. The server dynamically adjusts communication intervals and aggregates model parameters via a weighted average of individual agents' models to enhance robustness. By integrating adversarially robust models with on-device low-voltage learning, FortiSky improves security and energy efficiency tailored to the specific hardware.

Offline Generation of Adversarially Robust Model. The goal of offline robust model generation is to ensure generalized

Algorithm 2 FortiSky - adversarial and low-voltage bit-error robust learning framework for RL-based autonomous systems.

1: **Initialization:** number of agents n, communication interval CI, smoothing average threshold δ^k. For each agent, initialize action-value function Q with policy θ_i and target action-value function $\hat{Q}$ with policy $\theta^p = \theta$.
2: **for** *time step* $k = 1$ *to* T **do**
3: *// Agents conduct robust learning at each step*
4: **for** *each agent* i **in parallel do**
5: Update $\theta_i^k \leftarrow \texttt{RobustLearning}\left(i, \theta_i^{(k-1)}\right)$
6: **end for**
7: *// Agents communicate with server at every CI steps (multi-agent)*
8: **if** k *mod* $CI = 0$ **then**
9: Each agent i sends its policy θ_i^{k-} to server for smoothing average:
10: $\alpha^k = \frac{1}{n}\max(1, \frac{(1-n)k}{\delta^k} + n),\ \beta^k = \frac{1-\alpha^k}{n-1}$
11: **for** *each agent* i **do**
12: *Server* sends its updated policy θ_i^{k+} back to agent i:
13: $\theta_i^{k+} = \alpha^k \theta_i^{k-} + \beta^k \sum_{i \neq j} \theta_j^{k-}$
14: **end for**
15: **end if**
16: **end for**
17:
18: **Function:** `RobustLearning` $(i, \theta^{(k)})$
19: Given state s_k, take action a_k based on Q
20: Obtain reward r_k and reach new state s_{k+1}
21: Store transition (s_k, a_k, r_k, s_{k+1}) in D
22: *// Experience replay*
23: Sample a mini-batch $\{(s_j, a_j, r_j, s_{j+1})\}_{b=1}^B$ from D
24: *// Clean training pass*
25: Set $y_j = r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta^{p(k)})$
26: $\Delta^{(k)} = \nabla_\theta \sum_{b=1}^B (Q(s_j, a_j; \theta^{(k)}) - y_j)^2$
27: *// Perturbed training pass, inject adversarial errors with at most ϵ bit-flips (offline learning stage) or low-voltage bit errors at rate p (on-device learning stage)*
28: $\tilde{\theta}^{(k)} = \textsc{AdvBitErrors}_\epsilon(\theta^{(k)})$ (or $\tilde{\theta}^{(k)} = BErr_p(\theta^{(k)})$)
29: Set $\tilde{y_j} = (r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \tilde{\theta}^{p(k)}))$
30: $\tilde{\Delta}^{(k)} = \nabla_\theta \sum_{b=1}^B (Q(s_j, a_j; \tilde{\theta}^{(k)}) - \tilde{y_j})^2$
31: *// Average gradients and update w.r.t θ*
32: $\theta^{(k+1)} = \theta^{(k)} - \alpha(\Delta^{(k)} + \tilde{\Delta}^{(k)})$
33: *// Periodic update of target network*
34: Every C steps reset $\hat{Q} = Q$, i.e., set $\theta^p = \theta$
35: **Return** $\theta^{(k+1)}$
36:
37: **Output:** Unified adversarial and low-voltage random bit-error robust autonomy model θ =0

adversarial robustness and security across all UAV agents. The `RobustLearning` function in Algo. 2 describes FortiSky offline generation of robust RL algorithm with adversarial bit error injection. At each step k, the evaluation and target networks learn a policy $Q(\theta)$ by computing the predicted Q-value and gradient $\Delta^{(k)}$ with $\theta^{(k)}$ (line 23-27). Then FortiSky injects adversarial errors into the network (line 29) with the gradient-based adversarial bit error attack model in Sec. III-A and Algo. 1. To maintain accuracy under adversarial errors, the model is updated by averaging perturbed and unperturbed gradients (line 30–33). While the evaluation network θ is updated every step, the target network is updated every C steps by copying $\theta^{(k)}$ to $\theta_P^{(k)}$. This process yields robust models that navigate securely under adversarial attacks while mitigating low-voltage faults. In contrast, models trained only for low-voltage errors fail against adversarial attacks, since low-voltage faults are more frequent, random, and unstructured, unlike gradient-based adversarial errors.

On-Device Low-Voltage Robust Learning. After deploying the adversarially robust model onboard, agents can achieve additional efficiency by scaling down operating voltages. Al-

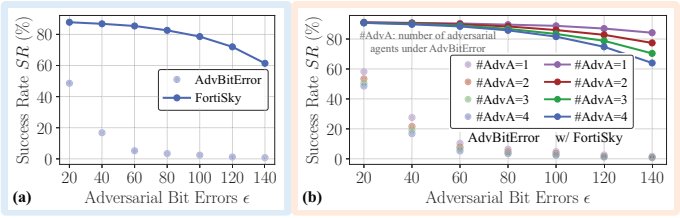


Fig. 5: Adversarial Robustness Improvement. FortiSky is evaluated on **(a)** single-UAV and **(b)** four-UAV swarm systems, and consistently improve success rate under AdvBitErrors. Multi-UAV systems exhibit higher adversarial robustness by virtue of collaborative learning.

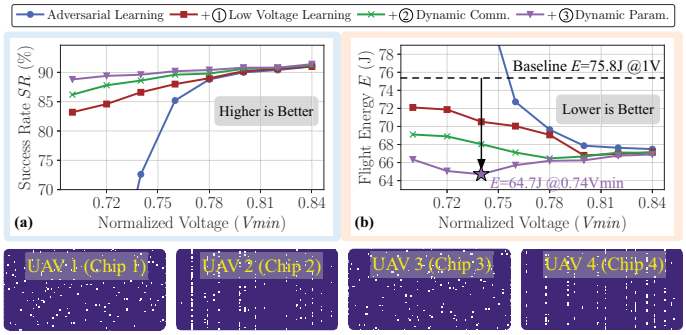


Fig. 6: Low-Voltage Bit-Error Robustness and Efficiency Improvements. FortiSky consistently improves robustness under low-voltage operations and AdvBitErrors (ϵ=40), and generalizes well across chips. Dynamic communication and agent-server optimizations further enhance mission success rate and energy efficiency.

though sub-threshold voltage introduces memory failures, FortiSky leverages its robust learning framework to adaptively learn and operate effectively under such conditions. Inspired by MulBERRY [23], FortiSky includes three on-device techniques:

① **Low-Voltage Bit Error Learning.** For an agent operating at low voltage during mission, it can adopt the unified `RobustLearning` function to learn using its own data on the low-voltage device incurring errors, and then perform the navigation task on the same hardware. Combining learning and navigation on the same device enables robust low-voltage operation with improved robustness tailored to specific chip. Reduced processing power enables further flight energy savings due to optimized UAV payload and enhanced agility [6].

② **Dynamic Communication Adjustment.** In multi-agent systems, each UAV communicates with the central server every CI steps. FortiSky dynamically adjusts the communication interval based on the UAVs' operating mode. UAVs running at low voltage with bit errors communicate with the server at half the normal frequency, facilitating faster adaptation to specific hardware conditions and limiting the effect of bit errors on the global model [23]. Lines 7-15 of Algo. 2 describe UAVs transmitting their autonomy model parameters at step k to the server, where k is a multiple of CI.

③ **Dynamic Server Parameters Adjustment.** The server aggregates model parameters θ_i^{k-} received from all UAVs and updates the global model via a weighted average computation. The updated model θ_i^{k+} is then distributed back to the UAVs for continued on-device robust learning (lines 10-15). FortiSky dynamically adjusts the weighting parameters α and β. Specifically, for UAVs operating under low-voltage conditions, the server assigns a doubled weight (2α) to emphasize learning the corresponding fault patterns, thereby enabling more targeted adaptation to chip-specific errors. After this adjustment, the system proceeds with the robust learning process as Algo. 2. In the absence of dynamic tuning, all weights default to $1/N$.

Beyond dynamic parameter adjustment, FortiSky adapts communication frequency and resilience mechanisms to handle agent failures (e.g., UAV malfunctions or delays). If a UAV fails, FortiSky reassigns its learning tasks to others or increases communication to maintain global progress.

	AdvBitErrors			LowVoltageErrors			AdvBitErrors + LowVoltageErrors			
	ϵ=20	ϵ=40	ϵ=60	0.78V_{min}	0.74V_{min}	0.70V_{min}	ϵ=20 / 0.78V_{min}	ϵ=20 / 0.74V_{min}	ϵ=60 / 0.78V_{min}	ϵ=60 / 0.74V_{min}
Baseline	48.6	26.8	15.2	87.6	67.6	24.2	46.6	39.0	13.4	9.8
Ⓐ AdvLearn	90.8	90.4	90.0	88.2	69.2	31.3	87.8	68.0	87.4	67.0
Ⓛ LowVoltageLearn	77.8	71.2	62.4	90.4	90.4	88.6	77.0	76.2	60.8	60.2
Ⓛ Offline+Ⓐ Offline	86.8	83.6	79.8	89.4	89.2	87.6	86.0	85.4	79.0	77.8
FortiSky	**90.8**	**90.4**	**90.0**	**90.2**	**90.2**	**88.6**	**90.4**	**90.0**	**90.2**	**88.2**

Success Rate (%) 90 75 60 45 30

Fig. 7: Correlation Between Adversarial and Low-Voltage Robust Learning. When Adversarial learning (AdvLearn) and Low-voltage learning (LowVoltageLearn) are applied individually, they are not capable of ensuring both adversarial robustness (AR) and bit-error robustness (BR). Similarly, a purely offline approach for learning both adversarial and low-voltage errors does not yield the desired mission success rate. In contrast, FortiSky, comprising of offline adversarial learning and on-device low voltage error learning, effectively strengthens both AR and BR and improves mission energy efficiency.

V. FortiSky Evaluation

This section evaluates FortiSky on UAV systems, demonstrating consistent improvements in both adversarial and low-voltage bit-error robustness. FortiSky enhances resilience and energy efficiency across single- and multi-UAV missions.

A. Experimental Setup

Simulation Platform. We utilize *PEDRA*, a real-flight-validated UAV infrastructure [35] for our evaluations. PEDRA integrates Unreal Engine [36] for environment simulation, AirSim [37] for capturing UAV dynamics and kinematics, and RL for generating real-time flight commands for each UAV.

Task and Policy Models. We study UAV navigation tasks (e.g., collaborative delivery) where UAVs travel from start to destination while avoiding obstacles. The action space A has 25 actions, with the C3F2 model (1.1 MB). Additional models and environments are shown in Fig. 8 and Tab. II.

UAV Platforms. We use DJI Tello [38], a UAV with 80g takeoff weight, 1100mAh battery and 13min max flight time. In Tab. II, we configure another UAV, DJI Spark [39] with 300g weight, 1480mAh battery and 16min max flight time.

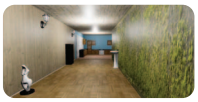
Evaluation Metrics. We measure robustness (success rate) and efficiency (processing and mission energy). Success rate is the percentage of flights reaching their destination, and mission energy is the average per flight. Each scenario averages 500 flights with a 1.3% error margin at 95% confidence.

B. Evaluation Results

Adversarial Robustness Improvement. We evaluate FortiSky on single- and four-UAV systems under adversarial errors (Sec. III-A). As shown in Fig. 5, FortiSky sustains >80%

TABLE I: Continuous Learning to Unseen Attacks. Mission success rate (%) under various attacks. In comparison to the baseline, FortiSky generalizes to both unseen L_p and non-L_p attacks.

		L_2		L_1		L_∞	
L_p-attacks	**Various L_p** (FortiSky trained offline with L_2)	FortiSky	w/o FortiSky	FortiSky	w/o FortiSky	FortiSky	w/o FortiSky
		90.4	18.2	86.0	17.4	85.8	12.6
	Various ϵ (FortiSky trained offline with $\epsilon = 40$)	$\epsilon = 40$		$\epsilon = 20$		$\epsilon = 60$	
		FortiSky	w/o FortiSky	FortiSky	w/o FortiSky	FortiSky	w/o FortiSky
		90.4	18.8	90.8	48.6	90.0	8.8
Non L_p-attacks		FortiSky		w/o FortiSky			
	Random policy attack	85.8		60.6			
	Opposite policy attack	84.2		52.2			


Sparse Obstacle


Medium Obstacle


Dense Obstacle

Environment		**Sparse Obstacle**		**Medium Obstacle**		**Dense Obstacle**	
Quality-of-Flight		Success Rate (%)	Flight Energy (J)	Success Rate (%)	Flight Energy (J)	Success Rate (%)	Flight Energy (J)
No Attack (ϵ=0 @$1V$)		93.2	62.6	90.6	75.8	88.8	98.0
Attack ϵ=40	w/o FortiSky	28.6	296.8	20.2	358.3	10.4	504.4
@$0.74V_{min}$	w/ FortiSky	93.2	53.1	90.4	64.7	88.6	88.1

Fig. 8: Effectiveness across Environments. FortiSky consistently improves mission robustness and efficiency under adversarial and low-voltage errors across various obstacle density environments.

mission success with ϵ=80 AdvBitError, while the baseline drops below 5%, matching prior results on tasks of similar difficulty [1], [25]. Multi-UAV systems further improve robustness (Fig. 5b). With one adversarial UAV, they consistently outperform single-UAV systems, and even with all UAVs adversarial, accuracy remains comparable. We hypothesize that collaborative learning contributes to enhanced robustness.

Low-Voltage Bit-Error Robustness and Efficiency Improvements. Fig. 6 shows that FortiSky improves robustness under simultaneous adversarial (ϵ=40) and low-voltage errors via offline and on-device learning. Dynamic communication and parameter tuning further boost success. Lowering voltage cuts processor temperature, eliminates heatsink payloads, and reduces energy: at $0.74V_{min}$, FortiSky saves 14.6% flight energy and 3.73× processing energy in multi-UAV systems under adversarial errors. Fig. 6 also shows generalization across chip-specific fault patterns when training is tailored per UAV.

Correlation Between Adversarial and Low-Voltage Robust Learning. Fig. 7 evaluates adversarial learning, low-voltage learning, and FortiSky under adversarial, low-voltage, and combined errors on a multi-UAV system. Adversarial and low-voltage learning each improve robustness to their target errors but offer little cross-robustness. Sequentially applying low-voltage then adversarial training is ineffective and further limited by UAV resource constraints. *By contrast, FortiSky integrates offline and on-device robust learning with dynamic knowledge sharing, improving both adversarial and bit-error robustness while enabling efficient low-voltage operation.*

Continuous Learning Against Unseen Attacks. We evaluate FortiSky's ability to adapt to unseen adversarial attacks at runtime (Tab. I). Trained offline with L_2 attacks, FortiSky achieves >85% success under L_1, L_2, and L_∞ attacks, and >90% across ϵ=20-60 (vs. <20% for baselines). It also maintains >84% success under non-L_p attacks (e.g., random and opposite-policy [18]). Under extreme cases (e.g., ϵ=160 L_2), success falls below 60%, showing robustness limits.

TABLE II: Effectiveness across UAVs and Models. FortiSky consistently improves mission robustness and efficiency under adversarial and low-voltage errors across a combination of UAVs and models.

UAVs / Models		**DJI Tello / C3F2**		**DJI Tello / C5F4**		**DJI Spark / C5F4**	
Quality-of-Flight		Success Rate (%)	Flight Energy (J)	Success Rate (%)	Flight Energy (J)	Success Rate (%)	Flight Energy (J)
No Attack (ϵ=0 @$1V$)		90.6	75.8	91.4	75.3	91.6	216.1
Attack ϵ=40	w/o FortiSky	20.2	358.3	26.8	344.5	26.6	1258.1
@$0.74V_{min}$	w/ FortiSky	90.4	64.7	91.4	64.2	91.4	194.4

TABLE III: Ablation Study on Communication Frequency. Success rate and energy under agent-server communication frequencies.

Communication Freq. ($1/CI$)	7/8	3/4	5/8	1/2	3/8	1/4	1/8
Success Rate (%)	87.6	89.8	89.4	90.4	90.2	88.6	87.2
Flight Energy (J)	69.1	67.4	64.9	64.7	65.5	66.1	69.7

Effectiveness across Environments. Fig. 8 evaluates FortiSky on three environments with different obstacle densities (sparse, medium and dense). We observe that FortiSky consistently improves robustness and is adaptive across environments under adversarial and bit errors, with 15.2%, 14.6%, and 10.1% flight energy reduction for three scenarios, respectively.

Effectiveness across UAVs and Models. Tab. II evaluates FortiSky on another UAV type (DJI Spark) and model C5F4 in a four-UAV system. DJI Spark has larger frame size than DJI Tello, and C5F4 has larger model size than C3F2. We observe that FortiSky consistently improves robustness and is adaptive across UAVs and models, with 14.6%, 14.7%, and 10.0% flight energy reduction for three UAV and model combinations.

Effectiveness in Larger-Scale UAV Deployment. We evaluate FortiSky on a larger-scale 12-UAV system and observe that FortiSky consistently improves adversarial and bit-error robustness, achieving 90.8% success rate as well as 13.8% mission energy reduction and 3.63× processing energy reduction under adversarial ϵ=40 and $0.75V_{min}$ operation.

Ablation Study on Optimization Parameters. We conduct the ablation studies on agent-server communication frequency and weighted-average parameters (Tab. III). We observe that for UAVs operating under low-voltage conditions, communicating at half of the normal frequency and doubling weighted factor results in optimal mission success rate and flight energy.

VI. Conclusion

This paper presented FortiSky, a unified robust learning framework that improves adversarial and low-voltage bit-error resilience for RL-based autonomous systems. By integrating offline adversarial training, on-device low-voltage-aware learning, and dynamic agent-server coordination, FortiSky enables reliable operation under aggressive voltage scaling and adversarial faults. Evaluations on single- and multi-UAV systems show near error-free mission success with significant processing and mission energy savings, demonstrating that robustness and efficiency can be co-optimized through cross-layer design.

Acknowledgements

This work was supported in part by CoCoSys, one of seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

REFERENCES

[1] B. Boroujerdian, H. Genc, S. Krishnan, W. Cui, A. Faust, and V. Reddi, "Mavbench: Micro aerial vehicle benchmarking," in *2018 51st annual IEEE/ACM international symposium on microarchitecture (MICRO)*, pp. 894–907, IEEE, 2018.

[2] F. Paredes-Vallés, J. J. Hagenaars, J. Dupeyroux, S. Stroobants, Y. Xu, and G. de Croon, "Fully neuromorphic vision and control for autonomous drone flight," *Science Robotics*, vol. 9, no. 90, p. eadi0591, 2024.

[3] T. van Dijk, C. De Wagter, and G. C. de Croon, "Visual route following for tiny autonomous robots," *Science Robotics*, vol. 9, no. 92, p. eadk0310, 2024.

[4] Z. Wan, Y. Du, M. Ibrahim, J. Qian, J. Jabbour, Y. Zhao, T. Krishna, A. Raychowdhury, and V. J. Reddi, "Reca: Integrated acceleration for real-time and efficient cooperative embodied autonomous agents," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Volume 2*, pp. 982–997, 2025.

[5] B. P. Duisterhof, S. Li, J. Burgués, V. J. Reddi, and G. C. de Croon, "Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9099–9106, IEEE, 2021.

[6] Z. Wan, N. Chandramoorthy, K. Swaminathan, P.-Y. Chen, V. J. Reddi, and A. Raychowdhury, "Berry: Bit error robustness for energy-efficient reinforcement learning-based autonomous systems," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2023.

[7] R. Hadidi, B. Asgari, S. Jijina, A. Amyette, N. Shoghi, and H. Kim, "Quantifying the design-space tradeoffs in autonomous drones," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 661–673, 2021.

[8] K. Roy, A. Kosta, T. Sharma, S. Negi, D. Sharma, U. Saxena, S. Roy, A. Raghunathan, Z. Wan, S. Spetalnick, *et al.*, "Breaking the memory wall: next-generation artificial intelligence hardware," *Frontiers in Science*, vol. 3, p. 1611658, 2025.

[9] P. Chen, M. Li, Z. Wan, Y.-S. Hsiao, M. Yu, V. J. Reddi, and Z. Liu, "Octocache: Caching voxels for accelerating 3d occupancy mapping in autonomous systems," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Volume 2*, pp. 704–718, 2025.

[10] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitraș, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 497–514, 2019.

[11] J. Tian, B. Wang, R. Guo, Z. Wang, K. Cao, and X. Wang, "Adversarial attacks and defenses for deep-learning-based unmanned aerial vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 22, pp. 22399–22409, 2021.

[12] Z. Wan, K. Swaminathan, P.-Y. Chen, N. Chandramoorthy, and A. Raychowdhury, "Analyzing and improving resilience and robustness of autonomous systems," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–9, 2022.

[13] S. Krishnan, Z. Wan, K. Bhardwaj, P. Whatmough, A. Faust, S. Neuman, G.-Y. Wei, D. Brooks, and V. J. Reddi, "Automatic domain-specific soc design for autonomous unmanned aerial vehicles," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 300–317, IEEE, 2022.

[14] S. Krishnan, Z. Wan, K. Bhardwaj, N. Jadhav, A. Faust, and V. J. Reddi, "Roofline model for uavs: A bottleneck analysis tool for onboard compute characterization of autonomous unmanned aerial vehicles," in *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 162–174, IEEE, 2022.

[15] N. Chandramoorthy, K. Swaminathan, M. Cochet, A. Paidimarri, S. Eldridge, R. V. Joshi, M. M. Ziegler, A. Buyuktosunoglu, and P. Bose, "Resilient low voltage accelerators for high energy efficiency," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 147–158, IEEE, 2019.

[16] D. Stutz, N. Chandramoorthy, M. Hein, and B. Schiele, "Bit error robustness for energy-efficient dnn accelerators," *Proceedings of Machine Learning and Systems (MLSys)*, vol. 3, pp. 569–598, 2021.

[17] S. Addepalli, S. Jain, *et al.*, "Efficient and effective augmentation strategy for adversarial training," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 1488–1501, 2022.

[18] A. Anwar and A. Raychowdhury, "Multi-task federated reinforcement learning with adversaries," *arXiv preprint arXiv:2103.06473*, 2021.

[19] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G.-Y. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 267–278, 2016.

[20] Z. Wan, A. Anwar, Y.-S. Hsiao, T. Jia, V. J. Reddi, and A. Raychowdhury, "Analyzing and improving fault tolerance of learning-based navigation systems," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 841–846, IEEE, 2021.

[21] Y.-S. Hsiao, Z. Wan, T. Jia, R. Ghosal, A. Mahmoud, A. Raychowdhury, D. Brooks, G.-Y. Wei, and V. J. Reddi, "Mavfi: An end-to-end fault analysis framework with anomaly detection and recovery for micro aerial vehicles," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2023.

[22] S. Kim, P. Howe, T. Moreau, A. Alaghi, L. Ceze, and V. Sathe, "Matic: Learning around errors for efficient low-voltage neural network accelerators," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2018.

[23] Z. Wan, N. Chandramoorthy, K. Swaminathan, P.-Y. Chen, K. Bhardwaj, V. J. Reddi, and A. Raychowdhury, "Mulberry: Enabling bit-error robustness for energy-efficient multi-agent autonomous systems," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Volume 2*, pp. 746–762, 2024.

[24] D. Stutz, N. Chandramoorthy, M. Hein, and B. Schiele, "Random and adversarial bit error robustness: Energy-efficient and secure dnn accelerators," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 45, no. 3, pp. 3632–3647, 2022.

[25] S. Krishnan, B. Boroujerdian, W. Fu, A. Faust, and V. J. Reddi, "Air learning: a deep reinforcement learning gym for autonomous aerial robot visual navigation," *Machine Learning*, vol. 110, no. 9, pp. 2501–2540, 2021.

[26] S. Zeng, M. A. Anwar, T. T. Doan, A. Raychowdhury, and J. Romberg, "A decentralized policy gradient approach to multi-task reinforcement learning," in *Uncertainty in Artificial Intelligence (UAI)*, pp. 1002–1012, PMLR, 2021.

[27] A. Panda, A. Baird, S. Pinisetty, and P. Roop, "Incremental security enforcement for cyber-physical systems," *IEEE Access*, vol. 11, pp. 18475–18498, 2023.

[28] O. Mutlu and J. S. Kim, "Rowhammer: A retrospective," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 8, pp. 1555–1571, 2019.

[29] F. Croce and M. Hein, "Provable robustness against all adversarial l_p-perturbations for p1.," in *International Conference on Learning Representations (ICLR)*, 2020.

[30] F. Croce and M. Hein, "Sparse and imperceivable adversarial attacks," in *Proceedings of the IEEE/CVF international conference on computer vision (CVPR)*, pp. 4724–4732, 2019.

[31] S. Ganapathy, J. Kalamatianos, K. Kasprak, and S. Raasch, "On characterizing near-threshold sram failures in finfet technology," in *Proceedings of the 54th Annual Design Automation Conference (DAC)*, pp. 1–6, 2017.

[32] S. Koppula, L. Orosa, A. G. Yağlıkçı, R. Azizi, T. Shahroodi, K. Kanellopoulos, and O. Mutlu, "Eden: Enabling energy-efficient, high-performance deep neural network inference using approximate dram," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 166–181, 2019.

[33] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pp. 1211–1220, 2019.

[34] A. Tang, S. Sethumadhavan, and S. Stolfo, "{CLKSCREW}: Exposing the perils of {Security-Oblivious} energy management," in *26th USENIX Security Symposium (USENIX Security 17)*, pp. 1057–1074, 2017.

[35] A. Anwar and A. Raychowdhury, "Autonomous navigation via deep reinforcement learning for resource constraint edge nodes using transfer learning," *IEEE Access*, vol. 8, pp. 26549–26560, 2020.

[36] W. Qiu and A. Yuille, "Unrealcv: Connecting computer vision to unreal engine," in *European Conference on Computer Vision (ECCV)*, pp. 909–916, Springer, 2016.

[37] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," pp. 621–635, Springer, 2018.

[38] "Dji ryze tech tello quadcopter." https://www.ryzerobotics.com/tello.

[39] "Dji spark specs." https://www.dji.com/spark/info.