MulBERRY: Enabling Bit-Error Robustness for Energy-Efficient Multi-Agent Autonomous Systems

Zishen Wan
zishenwan@gatech.edu
Georgia Institute of Technology
Atlanta, GA, USA

Nandhini Chandramoorthy
nandhini.chandramoorthy@ibm.com
IBM Research
Yorktown Heights, NY, USA

Karthik Swaminathan
kvswamin@us.ibm.com
IBM Research
Yorktown Heights, NY, USA

Pin-Yu Chen
pin-yu.chen@ibm.com
IBM Research
Yorktown Heights, NY, USA

Kshitij Bhardwaj
bhardwaj2@llnl.gov
Lawrence Livermore National Laboratory
Livermore, CA, USA

Vijay Janapa Reddi
vj@eecs.harvard.edu
Harvard University
Cambridge, MA, USA

Arijit Raychowdhury
arijit.raychowdhury@ece.gatech.edu
Georgia Institute of Technology
Atlanta, GA, USA

Abstract

The adoption of autonomous swarms, consisting of a multitude of unmanned aerial vehicles (UAVs), operating in a collaborative manner, has become prevalent in mainstream application domains for both military and civilian purposes. These swarms are expected to collaboratively carry out navigation tasks and employ complex reinforcement learning (RL) models within the stringent onboard size, weight, and power constraints. While techniques such as reducing onboard operating voltage can improve the energy efficiency of both computation and flight missions, they can lead to on-chip bit failures that are detrimental to mission safety and performance.

To this end, we propose MulBERRY, a multi-agent robust learning framework to enhance bit error robustness and energy efficiency for resource-constrained autonomous UAV swarms. MulBERRY supports multi-agent robust learning, both offline and on-device, with adaptive and collaborative agent-server optimizations. For the first time, MulBERRY demonstrates the practicality of robust low-voltage operation on multi-UAV systems leading to energy savings in both compute and mission quality-of-flight. We conduct extensive system-level experiments on autonomous multi-UAV navigation by leveraging algorithm-level robust learning techniques, and hardware-level bit error, thermal, and power characterizations. Through evaluations, we demonstrate that MulBERRY achieves robustness-performance-efficiency co-optimizations for autonomous UAV swarms. We also show that MulBERRY generalizes well across fault patterns, environments, UAV types, UAV agent numbers, and RL policies, with up to 18.97% reduction in flight energy, 22.07% increase in the number of successful missions, and 4.16× processing energy reduction.

ACM Reference Format:
Zishen Wan, Nandhini Chandramoorthy, Karthik Swaminathan, Pin-Yu Chen, Kshitij Bhardwaj, Vijay Janapa Reddi, and Arijit Raychowdhury. 2024. MulBERRY: Enabling Bit-Error Robustness for Energy-Efficient Multi-Agent Autonomous Systems. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24), April 27-May 1, 2024, La Jolla, CA, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3620665.3640420

ASPLOS '24, April 27-May 1, 2024, La Jolla, CA, USA

ACM ISBN 979-8-4007-0385-0/24/04
https://doi.org/10.1145/3620665.3640420

1 Introduction

The use of autonomous Unmanned Aerial Vehicles (UAVs) swarms is becoming increasingly prevalent in mainstream application domains such as surveillance, monitoring, transportation, package delivery, and search-and-rescue operations [1, 52, 69, 70]. To achieve the level of autonomy required for these applications, swarms of UAVs typically collaboratively execute complex reinforcement learning (RL) models on-board across complex surrounding environments to finish the missions, with little-to-no offloading computation support [17, 22]. Moreover, when deployed in harsh physical conditions or hostile environments, as is often the case in military and civilian scenarios, such swarms can ensure sustained operation even when individual UAVs are lost during missions.

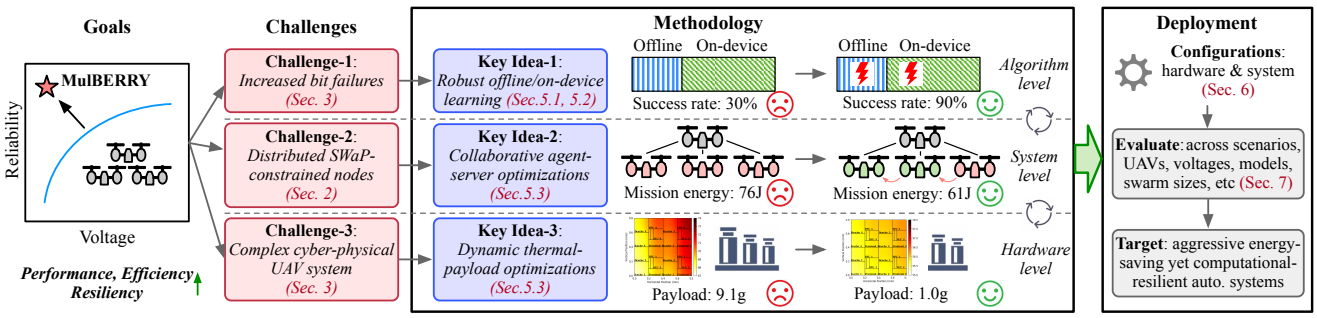


Figure 1. MulBERRY System Overview. MulBERRY is a robust learning framework with the goal to achieve aggressive energy-savings yet computational-resilient autonomous swarms. MulBERRY addresses the challenges of increased low-voltage induced bit failures, distributed resource-constrained nodes, and complex cyber-physical UAV systems, by proposing methodologies including robust offline/on-device learning, collaborative agent-server optimizations, and dynamic thermal-payload optimizations. MulBERRY is deployed across hardware, system, and application scenarios and consistently demonstrates performance-efficiency-robustness improvements for multi-UAV systems.

However, these distributed safety-critical UAVs are typically Size, Weight, and Power (SWaP) constrained [21], making it imperative to maximize their processing efficiency without compromising on mission robustness and safety [68]. As a result, several energy-saving optimizations [51, 61] have to be applied to ensure that both mechanical and processing components can successfully complete within the specified SWaP constraints.

Lowering the supply voltage of the onboard processor is an effective means of improving its energy efficiency, given the quadratic relation between energy and operating voltage. Although the processing power is only a small fraction of the total UAV power with the majority being used for flight motion, a small reduction in processing power would enable it to be re-targeted towards increasing the safe flight speed, thus resulting in significant energy savings due to reduced flight time [9, 36].

However, aggressively scaling down the voltage towards near-threshold ranges can have adverse implications on the overall reliability of UAV systems. Operating below rated voltage ranges can result in memory bit failures [15, 24, 31, 39, 56] and logic timing errors [74, 75]. Consequently, autonomous swarms that operate in the presence of errors will experience a drop in task success rate and an increase in mission time due to path detours or battery depletion caused by incorrect navigation. As a result, flight energy may actually increase despite lower voltage (Sec. 4).

While reducing the operating frequency can help mitigate logic timing errors at low voltages, the on-chip SRAMs are vulnerable to read disturbs which can result in bit errors even at reduced frequencies [38]. Recent works have proposed techniques to mitigate the effects of memory bit errors, including circuit/architecture co-optimizations [15, 30] and algorithmic techniques such as error-aware training to realize DNN models that are resilient to bit errors [30, 31, 60]. However, they either incur area and power overheads for SWaP-constrained systems or are not applicable to multi-UAV scenarios. Although modern-day processors are equipped with RAS-enhancing features such as parity, ECC, and redundancy, they are primarily targeted toward mitigating transient faults that usually result in single/double bit errors, and are ineffective against low-voltage induced errors that are inherently multi-bit in nature.

To this end, in this paper, we present MulBERRY, a multi-agent robust learning framework that optimizes the processing efficiency, mission reliability, and overall energy efficiency for autonomous UAV swarms (Fig. 1). MulBERRY comprises of a cross-layer framework that integrates *algorithm-level* error-aware learning techniques with *system-level* collaborative server-agent optimizations and *hardware-level* thermal-voltage adaptive adjustments. MulBERRY achieves aggressive energy-savings yet computational-resilient autonomous swarms, with substantial improvements in end-to-end robustness and efficiency (e.g., mission success rate and quality-of-flights) compared to state-of-the-art architectural and algorithmic techniques for enabling robust low-voltage operation.

MulBERRY supports both offline and on-device multi-agent robust learning paradigms to enable reliable execution in the presence of bit flips at low voltages (Sec. 4). MulBERRY applies offline bit error injection techniques, similar to those proposed in [31, 60, 65], to a multi-agent scenario and generates RL models robust to low voltage-induced bit errors. In addition, directly learning on low-voltage devices with errors affecting parameters further enables UAVs to reliably operate at lower voltages and tolerate higher bit error rates, further improving mission success rates and higher flight energy savings tailored to each UAV chip.

MulBERRY leverages the unique characteristics of multi-agent learning and proposes several collaborative agent-server optimizations (Sec. 5). Inspired by the computational sprinting technique in mobile processors [53, 54], UAVs engage in collaborative *sprint-or-slack* operations where agents can alternate between *slacking* at low and *sprinting* at high voltages and share model updates, thus simultaneously enabling robust yet energy-efficient execution for time intervals. Considering the correlation between payload and UAV agility [9, 36, 65], MulBERRY can also reduce mechanical energy through workload-aware payload optimization. With dynamic agent-server communication and weighted aggregation adjustments, MulBERRY further significantly improves the performance-efficiency-robustness of autonomous UAV swarms.

We evaluate MulBERRY across several SWaP-constrained multi-UAV autonomous navigation scenarios (Sec. 6, Sec. 7), and demonstrate that MulBERRY is compatible with low-power processors and algorithms, and generalizes well across environments, UAV types, operating voltages, RL models, mission tasks, number of UAVs, and chip bit error patterns, with consistent robustness, efficiency, and mission performance improvements.

This paper, therefore, makes the following contributions:

- We systematically analyze the impact of low-voltage processing on autonomous UAV swarms in terms of their energy efficiency and mission performance.
- We propose a host of optimization techniques for robust and efficient multi-agent systems. These include agent-side payload optimization through collaborative sprint-or-slack operation, and server-side dynamic communication and adaptive knowledge-sharing, each supported by detailed robustness, energy, and performance characterizations.
- We demonstrate a novel sprint-or-slack methodology that minimizes the average power and temperature across all UAVs with detailed technology-level power and thermal models. This, in turn, enables payload optimization, thus significantly improving the compute-mechanical system-level efficiency.
- We present MulBERRY, a framework that supports collaborative robust learning in autonomous UAV swarms, both offline and on-device, to improve *processing efficiency*, *task robustness*, and *mission performance*.
- We extensively evaluate MulBERRY on numerous multi-UAV scenarios through cross-layer silicon-to-application analysis, and show that MulBERRY generalizes across voltage ranges, error patterns, RL models, environments, and SWaP-constrained UAV types, achieving up to 18.97% energy savings, 22.07% increase in successful missions with 4.16× processing energy reduction on multi-UAV tasks compared to an equivalent system operating at the baseline voltage of 1V.

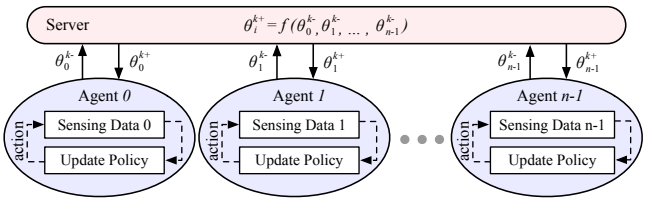


Figure 2. Multi-Agent Autonomous System. Agents jointly learn a unified model by interacting with environments to achieve the defined goals. Agents communicate with the server without sharing local data.

2 Multi-agent Autonomous Systems: An Overview

This section formally introduces the multi-agent autonomous system, serving as the focal point and baseline in this paper. In multi-agent autonomous systems, the agents aim to jointly learn a unified model by interacting with environments to achieve defined goals. Each agent acts and makes observations in its own environment, and shares the information with a centralized server. Each agent adopts an RL model to interact with the environment which can be characterized by a Markov decision process (MDP).

The MDP at each agent i can be modeled as a tuple $M_i = (S_i, A_i, P_i, R_i, \gamma_i)$, where S_i is the state space, A_i is the action space, P_i is the MDP transition probabilities, $R_i : S_i \times A_i \rightarrow \mathbb{R}$ is the reward function, and $\gamma_i \in (0, 1)$ is the discount factor.

Let V_i^π be the value function at the state s of agent i induced by the model π. With $a_i^k \sim \pi(\cdot|s_i^k)$, we have:

$$V_i^\pi(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma_i^k R_i(s_i^k, a_i^k) \mid s_i^0 = s\right] \tag{1}$$

The goal of the multi-agent autonomous system is to learn a unified model π^* that maximizes the sum of long-term return, quantified by the value function $V_i^\pi(s)$ for all agents:

$$\max_\pi V(\pi; \boldsymbol{\rho}) \triangleq \sum_{i=0}^{n-1} \mathbb{E}_{s_i \sim \rho_i} V_i^\pi(s_i) \tag{2}$$

where $\boldsymbol{\rho} = [\rho_0, \ldots, \rho_{n-1}]^T$ and ρ_i denotes the initial state distribution over A_i of agent i. Solving Eq. 2 yields a unified π^* resulting in a balanced performance across n agents.

We use θ to model the family of policies $\pi_\theta(a|s)$, then the goal of the multi-agent problem is to find θ^* that satisfies:

$$\theta^* = \arg\max_\theta V(\theta; \boldsymbol{\rho}) \triangleq \sum_{i=0}^{n-1} \mathbb{E}_{s_i \sim \rho_i} V_i^{\pi_\theta}(s_i) \tag{3}$$

To find the unified model in Eq. 2, in multi-agent systems, M_i remains at the local agent i while the model θ_i is shared with a designated server (Fig. 2). Each agent i learns its own task by utilizing its local data $\mathcal{D}_i$ to train θ_i. After completing k steps, agents share their model θ_i^{k-} with the server. The server carries out a weighted aggregate and generates n new

Algorithm 1 Multi-agent autonomous system (*MA*)

Initialization: number of agent 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 $\theta^p = \theta$
for *time step* $k = 1$ *to* T **do**
  // Each agents interact with its own environment and update the policy
  for *each agent* i **in parallel do**
    Update $\theta_i^{k-} \leftarrow \texttt{AgentUpdate}\left(i, \theta_i^{(k-1)+}\right)$
  end for
  // Agents communicate with server at every CI steps
  if k *mod* $CI = 0$ **then**
    Each agent i sends its policy θ_i^{k-} to server
    Server calculates smoothing average parameters:
    $\alpha^k = \frac{1}{n}\max(1, \frac{(1-n)k}{\delta^k} + n),\ \beta^k = \frac{1-\alpha^k}{n-1}$
    for *each agent* i **do**
      Server sends updated policy θ_i^{k+} back to agent i:
      $\theta_i^{k+} = \alpha^k \theta_i^{k-} + \beta^k \sum_{i \neq j} \theta_j^{k-}$
    end for
  end if
end for

Function: `AgentUpdate` (i, θ)
1) Compute the gradient of the local value function $\frac{\partial V_i^{\pi_\theta}(\rho_i)}{\partial \theta_{s_i,a_i}}$ based on
the local data
2) Update the policy parameter $\theta^- = \theta + \delta^k \frac{\partial V_i^{\pi_\theta}(\rho_i)}{\partial \theta_{s_i,a_i}}$
Return θ^-

Output: Unified multi-agent autonomous system (*MA*) policy θ

sets of parameters θ_i^{k+}, one for each agent, using:

$$\theta_i^{k+} = \alpha^k \theta_i^{k-} + \beta^k \sum_{j \neq i} \theta_j^{k-} \tag{4}$$

where $\alpha^k, \beta^k = \frac{1-\alpha}{n-1} \in (0, 1)$ 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 training proceeds, the weighted aggregate constants converge to $\alpha^k, \beta^k \to \frac{1}{n}$. The conditions on α^k, β^k to guarantee the convergence can be found in [73]. The completed algorithm of the multi-agent system is listed in Algo. 1.

In this work, we consider a multi-UAV autonomous system, where each UAV interacts with its own sensor data and updates local model parameters. Each UAV shares its model with the central server, and the server computes a weighted aggregate and finally all UAVs converge to a unified model. The multi-UAV system is first trained offline on meta environments (offline learning), and then the knowledge is transferred to the real environment with fine-tuned (on-device learning).

3 Low-Voltage Induced Bit Errors

This section first introduces the challenges of low-voltage operation and UAV system implications (Sec. 3.1), then presents the abstracted error model used in our framework (Sec. 3.2).

3.1 Low-Voltage Bit Error Characterization

Low-Voltage Operation and Energy Lowering operating voltages towards near-threshold ranges intensifies bit cell variations. This manifests as an exponential increase in bit error rates (BERs), adversely affecting accelerator memories where weights are stored. Fig. 3 shows this relationship from an exemplary FinFET SRAM chip fabricated in [15] and a sample spatial distribution of bit errors in a segment of the tested memory arrays. At a given voltage, these bit flips are persistent across multiple reads and writes to the same location. The locations are random and independent of each other across different chips and arrays [15, 24, 30, 32, 60]. Note that these voltage-induced errors are not transient and cannot be mitigated through spatial and temporal computational redundancy. Similarly, standard ECC may not correct all errors since there could be multiple faulty bits per memory word.

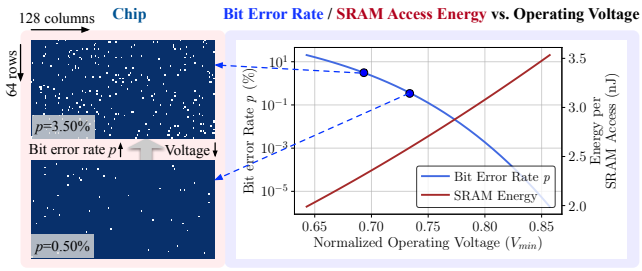


Figure 3. Low-Voltage Operation, Bit Errors and Energy in Memory. SRAM bit error rate p (blue, left y-axis) increases and energy (red, right y-axis) drops with decreasing operating voltage from 14nm FinFET SRAM arrays, as described in [15]. The voltage (x-axis) is normalized to V_{min}, the lowest measured voltage where there are no bit errors. Reproduced on the left is a random spatial error pattern in a cross-section of the memory array from [15, 60].

Frequency Scaling and UAV Implication. SRAMs are vulnerable to read disturbance failures as we scale down the voltage aggressively and these are distinct from timing errors. As illustrated in Fig. 4a, as we lower the voltage, the first memory bit failures are observed at $V_{1st-error(mem)}$ and the first logic timing errors are observed at $V_{1st-error(comp)}$ as we reduce the frequency along with voltage scaling. MulBERRY exploits this range ΔV between the lowest possible voltage for memory ($V_{1st-error(comp)}$) and logic ($V_{1st-error(mem)}$) with frequency scaling range ΔF. Within ΔV and ΔF, low voltage operation only results in read disturbances that are pertinent to SRAMs, while logic and register files are free from timing errors.

It is worth noting that as we scale the frequency with lower voltage, the functioning of UAVs remains unaffected. To verify this, we adopt a UAV roofline analysis model [35, 37]. The UAV, being a cyber-physical system, consists of sensors, onboard compute, and a flight controller (Fig. 4b). Thus, the

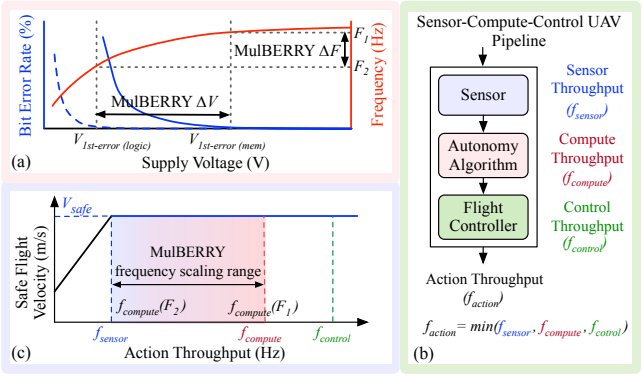


Figure 4. Frequency Scaling and UAV Implication. (a) MulBERRY exploits the voltage range between $V_{1st-error}$ at which first memory bit failures and first logic timing errors begin to happen. **(b)** UAV is a cyber-physical system composed of sensors, compute, and flight controller. **(c)** As we scale frequency with lower voltage, we ensure the logic is timing error-free and the functioning of UAVs remains unaffected, which is validated with UAV roofline model [35].

roofline model is to determine which of the UAV components limits the safe operating velocity. Higher safe velocity ensures that the UAV is reactive to dynamic environments and ensures that the UAV finishes tasks quickly, thereby lowering mission time and energy. As shown in Fig. 4c, we ensure that the decision-making rate (action throughput) is not bounded by onboard compute during frequency scaling, thus preserving UAV performance.

3.2 Bit Error Model

We extract the random fault model from silicon characterizations (Sec. 3.1). Bit flips to preferred states of 0 or 1 happen at a given voltage, and are persistent across multiple reads and writes at a given voltage. For a given memory array, given bit errors also persist across lower supply voltages, i.e., when a bit is erroneous at a given voltage it is also faulty at a lower voltage as observed in Fig. 3. Across different SRAM arrays, the patterns or spatial distribution of bit errors is usually different and can be assumed random [15, 24, 30].

At a given bit error rate p corresponding to a particular voltage on the voltage vs BER curves, we generate a uniform random distribution of bit flips, half of which flip to 0 and half of which flip to 1. Thus, for W weights with m bits per value, we sample uniformly $u \sim U(0,1)^{W \times m}$. The j-th bit in weight W_i is flipped if $u_{ij} \leq p$. We apply per-layer 8-bit symmetric quantization with 2's complement representation, and the quantized model weights are mapped linearly to the memory, without requiring knowledge of the exact bit error spatial distributions and hardware or application change.

4 Robustness of Multi-Agent Systems

This section explores how memory bit errors due to low-voltage operation impact the robustness and quality-of-flight metrics of multi-agent systems (Sec. 4.1) and then compares the mission performance of multi- and single-agent systems to elaborate their robustness characteristics (Sec. 4.2).

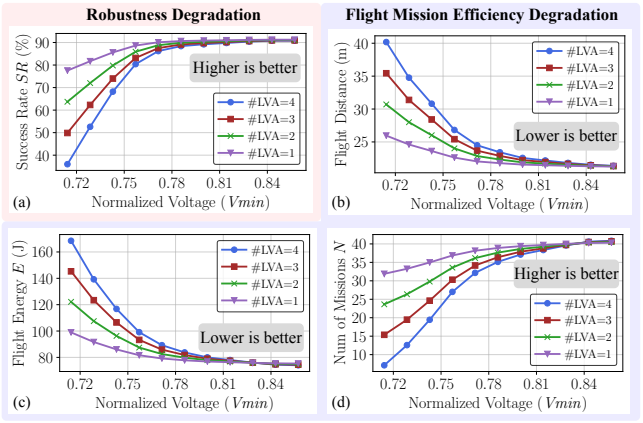


Figure 5. Fault Impact on Multi-UAV Systems. Low voltage-induced bit errors degrade the average success rate and quality-of-flight of a four-UAV system, exacerbated when more UAVs operate at low voltages. #LVA indicates the number of low-voltage UAVs.

4.1 Fault Impact on Multi-UAV Quality-of-Flight

We evaluate the impact of low-voltage operation on a multi-agent system where four UAVs aim to collaboratively navigate from start to goals in the shortest time without colliding with obstacles (Sec. 6). Fig. 5 shows the average task success rate and quality-of-flight across UAVs during the mission.

Task Success Rate: Fig. 5a illustrates that the low-voltage induced bit failures greatly degrade multi-UAV system robustness, exacerbated when more agents operate at low voltages. The increasing errors corrupt the error-free model and result in agents taking incorrect actions that compromise mission success. For example, the task success rate for a four-UAV system drops to 80% from 91% when voltage is reduced to $0.76V_{min}$ on account of adverse effects on the following parameters.

Flight Time: The corrupted flight actions due to bit failures can lead to path detours, resulting in longer trajectories and extended flight time on average for a given task (Fig. 5b).

Flight Energy: The increased flight time increases mission energy despite the processing energy savings from low-voltage operation, since the majority of flight energy is consumed by rotors that closely correlates with flight time (Fig. 5c).

Endurance: The increased flight energy and duration further reduces the number of missions that each agent can complete on a single charge before its battery depletes (Fig. 5d).

4.2 Comparison of Single/Multi-UAV Systems

We compare multi-agent and single-agent systems in terms of their overall robustness to low-voltage errors. Multi-agent systems leverage the central server to learn a shared model

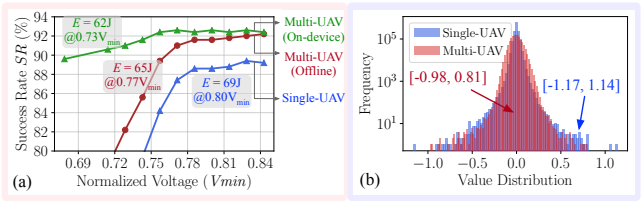


Figure 6. Multi-UAV and Single-UAV Comparison. Model trained in multi-UAV exhibits higher robustness and mission performance than single-UAV model, and exhibits narrower weight distribution.

collaboratively. Each agent learns a local model and communicates with the server, and in turn receives an updated model. The server computes the weighted aggregate of all received models from agents. Each agent is fitted with its own sensor and captures data from different perspectives of the environment. In contrast, a single-agent system learns only using its own data. Offline and on-device MulBERRY (Sec. 5) is applied in both scenarios.

In general, multi-agent systems exhibit better mission performance and robustness in terms of mission success rate compared to a single-agent model (Fig. 6a). Three out of four agents in Fig. 6a operate at low voltage. It is evident from the offline training scenario that the voltage at which the mission success rate drops precipitously is lower in the multi-agent system as compared to the single-agent system. Further learning on-device enables lowering operating voltage even further with a maintained mission success rate. We hypothesize that in multi-agent systems, models from other agents learned on their own local sensor data during training shared via the central server could contribute to higher robustness. Interestingly, Fig. 6b demonstrates the model learned in multi-agent systems has a narrower weight distribution than single-agent. We hypothesize that at the same BER, bit errors are distributed across more weights contributing to the final large logits with high confidence, therefore the errors incurred by bit flips are naturally minimized [60].

5 MulBERRY Framework

This section first presents the overview of MulBERRY multi-agent robust learning framework (Sec. 5.1), then describes how MulBERRY supports both offline (Sec. 5.2) and on-device robust learning (Sec. 5.3), with a host of dynamic and collaborative agent-server optimization techniques to achieve resiliency, performance, and efficiency improvements.

5.1 Overview

Fig. 7 shows the MulBERRY framework for robust learning in an autonomous swarm of UAVs. In this framework, individual UAV agents collaboratively solve navigation tasks with a server with the help of offline and on-device learning techniques to mitigate the effects of low voltage-induced bit errors.

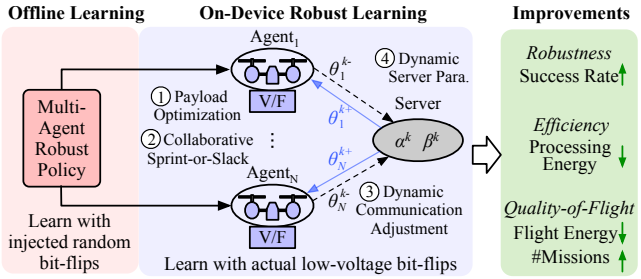


Figure 7. MulBERRY Multi-Agent Robust Learning. MulBERRY supports offline and on-device robust learning, with injected bit errors, payload optimization, collaborative sprinting, dynamic communication and server adjustment, for higher robustness, efficiency, and quality-of-flight.

Offline Robust Learning. MulBERRY first conducts a round of offline training through multi-agent RL with random bit error injection at nominal voltage (Sec. 5.2). Then each agent is equipped with this offline learned robust model. For an agent with no learning capabilities, the navigation task and flight actions are performed in inference-only mode using the offline robust model. This is effective up to a certain voltage, below which the task will fail at high BERs (Fig. 6a).

On-Device Robust Learning. As voltage scales down further, learning on-device is necessary to tolerate high BERs. MulBERRY supports collaborative on-device learning for all agents that communicate through a central server. Each agent learns using its own data on the low-voltage device incurring errors, and then performs the navigation task on the same hardware. Each agent communicates with the server at certain periodic intervals. The server updates the model with a weighted average of parameters with the capability to dynamically adjust those weights, then sends the updated model back to each agent who continues learning on-device. The agent is equipped with the capability to dynamically adjust voltage and frequency with *sprint-or-slack* mode of operation that it can sprint at error-free nominal voltage or slack at low-voltage with bit errors (Sec. 5.3). Combining learning and navigation on the same device ensures improved success rates and higher flight energy savings tailored to the specific chip.

5.2 Offline Generation of Robust Multi-Agent Model

The goal of generating robust policy offline is to provide generalized robustness across all agents, independent of whether they have the ability to learn on the device. Algo. 2 (`BitFlipLearning` function) shows our proposed algorithm for the generation of a bit-error robust RL model, by means of offline 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 respect to $\theta^{(k)}$ (line 23-27). Then, bit errors are injected into the network as would be the case when deployed on low-voltage devices (line 29). Random spatial bit error patterns are generated

Algorithm 2 MulBERRY - bit error robust learning framework for reinforcement learning-based multi-agent autonomous UAV systems

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

with bits equally flipping from 1-to-0 and 0-to-1. The parameters are quantized to 8 bits with rounding, and bit errors are applied to $\theta^{(k)}$ and $\theta_P^{(k)}$. The model is updated using an average of perturbed and unperturbed gradients so as to retain accuracy both with and without bit errors (lines 30-33). While the evaluation network θ is updated every step, the target network is updated every C step to keep training stable by copying over $\theta^{(k)}$ to $\theta_P^{(k)}$.

This method generates a model trained with bit error injection to simulate low-voltage device deployments, leading to robust navigation across agents with various bit error patterns, up to a certain voltage without further learning. Below that, when BER exceeds a threshold, the navigation success rate rapidly declines necessitating on-device training on each agent tailored for its own bit error pattern.

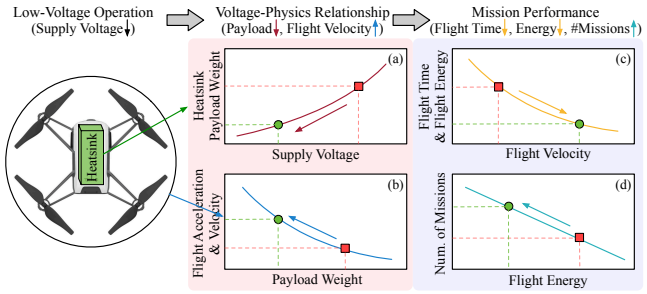


Figure 8. Low-Voltage Payload Optimization. Relation between supply voltage, payload weight, velocity, flight time, flight energy, and the number of missions [65]. **(a)** Reducing the voltage of the onboard processing unit helps reduce peak temperature, and correspondingly, the heatsink size and weight. **(b)** A decrease in payload weight can improve UAVs' acceleration and velocity, thus **(c)** reducing flight time and energy, **(d)** making the UAV complete more number of missions on a single battery charge.

5.3 On-Device Multi-Agent Robust Learning

Agents with on-device learning capability can achieve higher energy efficiency by further scaling down the operating voltage. Algo. 2 describes MulBERRY supporting multi-agent on-device learning. Each agent learns with its own data and is equipped with a bag of circuit/architecture methods to dynamically adjust the operating voltage to at least two levels (nominal and low), enabling it to learn in the presence of bit errors at low voltage using `BitFlipLearning` at a given step (line 5).

① **Low-Voltage Payload Optimization.** Lowering operating voltage has quadratic savings on power, but leads to an increase in learning steps and therefore flight time. However, as illustrated in [9, 36, 65], lowering the payload of the agent UAV additionally leads to increased acceleration and velocity. The enhanced flight velocity further reduces the time for the UAV to move at each step, thereby having a compensatory effect on flight time (Fig. 8). The UAV payload can be decreased by reducing the heatsink size, a concomitant positive effect of low-voltage operation. Therefore, MulBERRY proposes to tailor the UAV heatsink size to suit the lowered peak temperature corresponding to low voltage.

② **Collaborative Sprint-or-Slack Operation.** Computational sprinting techniques have been previously proposed to maximize the performance of mobile processors under power/thermal constraints [53, 54]. In a similar vein, we propose a collaborative sprint-or-slack operation scheme for efficient and resilient swarm system design. When an agent operates at nominal voltage, it *sprints* - the learning takes place without errors and therefore takes fewer steps to navigate to the destination. When an agent operates at low voltage, it *slacks* - the learning takes place with low voltage-induced memory bit errors causing it to go off-course, and thus take more time steps to complete. Although the frequency is also lower when it slacks, the computation latency

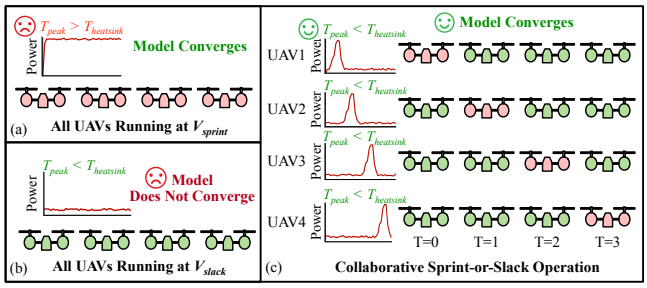


Figure 9. Collaborative Sprint-or-Slack Operation Mode. (a) When all UAVs run at V_{sprint}, the peak temperature may exceed the permissible value, while in **(b)** the model fails when all UAVs run at V_{slack}. By contrast, **(c)** collaborative *Sprint-or-Slack* framework ensures model convergence within permissible thermal limits.

in both modes is still less than the sensor latency that captures the input and the real-time processing requirement is still satisfied.

The sprint-or-slack operation ensures that the peak temperature limit is never breached (Fig. 9). When all agents sprint, flight time is lower, but flight energy increases due to increased power. When all agents slack, flight time is high due to an increase in the number of steps, and the collaborative learning is not effective when all agents are faulty. However, with each agent sprinting for a few short time steps in non-overlapping intervals, the success rate is higher and flight energy is lower. In addition, computation stays within thermal limits dictated by a heatsink configured for low-voltage operation. Thus, smaller heatsinks, coupled with short sprinting intervals in a round-robin fashion amongst agents keep the flight duration low and flight velocity high. Thus the overall flight energy is lowered and robustness is increased, as shown in Sec. 7.

③ **Dynamic Communication Adjustment.** MulBERRY proposes to allow dynamic server-agent communication and knowledge-sharing parameter adjustments in swarm systems. Agents communicate with the server every *CI* step and can be adjusted dynamically based on *sprint-or-slack* operating mode. A sprinting agent communicates with double the frequency compared to a slacking agent, leading to collaborative learning with increased robustness. Lines 8 and 9 indicate each agent sending its network parameters $\theta^{(k-)}$ at time step k, where k is a multiple of *CI*, to the server.

④ **Dynamic Server Parameters Adjustment.** The server receives network parameters $\theta_1^{(k-)}, \ldots, \theta_N^{(k-)}$ from all agents and computes updated parameters $\theta_1^{(k+)}, \ldots, \theta_N^{(k+)}$ for each. The server computes the parameters for agent i as a weighted average of parameters from all other agents, where the weights (line 11) are adjusted according to an agent sprinting or slacking. A slacking agent learns its bit error pattern, thus the server weights its parameters the same (0.4) as the sprinting agent (0.4) for its updated parameters, while all other slacking agents are weighted equally and sum to 0.2. After this dynamic knowledge-sharing adjustment, the system then follows Algo. 2 for subsequent robust learning processes. With no dynamic adjustment, the weights are simply equal $(1/N)$.

Through the proposed dynamic communication frequency and server parameter adjustment schemes, MulBERRY is also able to handle failures of individual agents (e.g., the UAV not functioning well or facing communication error/delay). For example, for non-functioning UAVs, MulBERRY can reassign the learning tasks across other UAVs by leveraging the aforementioned multi-agent learning paradigms and/or increasing communication frequency.

6 Experimental Setup

This section presents MulBERRY evaluation setup, including the system-level UAV platform and model (Sec. 6.1), and silicon-level compute and sprinting configurations (Sec. 6.2).

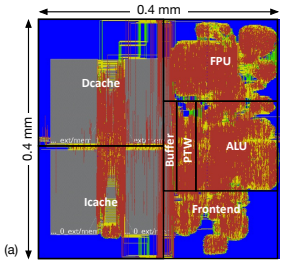
6.1 System-Level UAV Experimental Setup

Simulation Platform. We build upon PEDRA, an open-source multi-UAV simulation infrastructure [2], for MulBERRY evaluations. PEDRA incorporates Unreal Engine for environment simulation, AirSim for capturing UAV dynamics and kinematics, and multi-agent RL algorithms for generating real-time flight commands for multi-UAV scenarios. We use the same UAV energy and battery model as MAVBench [9] and Air Learning simulator [33].

Task and Policy Models. We focus on the multi-agent autonomous navigation task (e.g., collaborative package delivery or surveillance) where the UAVs start from a designated location and navigate through the environment to their destinations without colliding with obstacles. We use a perception-based probabilistic action space A with 25 actions, and the C3F2 neural network policy (3 Conv layers and 2 FC layers) with 1.1MB parameters [63]. We also evaluate more complex policies in Sec. 7.3. We focus on a 4-UAV scenario for MulBERRY evaluation and analyze a 12-UAV scenario in Sec. 7.2.

UAV Platforms. We use Crazyflie nano UAVs [8], with $27g$ takeoff weight, $15g$ max payload, $250mAh$ battery capacity, and 7min max flight time. In Sec. 7.3, we configure two other UAVs: one is DJI Tellos [19] with $80g$ takeoff weight, $1100mAh$ battery and $13min$ max flight time; another is DJI Spark [20] with $300g$ takeoff weight, $1480mAh$ battery and $16min$ max flight time for evaluation.

Evaluation Metrics. We evaluate both compute-level efficiency (processing energy) and mission-level metrics (average success rate, mission completion time, energy, and number of missions). Success rate (*SR*) is the percentage of successful trials. A successful mission is achieved when the UAV swarm safely reaches its destination before running out of battery. Mission completion time and energy (E') correspond to the average time and energy required for UAVs to reach goals in a single mission. The number of missions (N)



(b)

Hardware Configuration Parameters	
Technology	GF 12*nm*
Core Type	4 x RISC-V Rocket Cores
Cache	16KB 4-way I+D Caches
Routed Core Area	0.4*mm* x 0.4*mm*
Voltage Range	0.54*V* to 1*V*
Power	117*mW* to 399*mW*

Figure 10. UAV Compute Hardware. (a) Post-PnR layout of the RISC-V processing core used for each UAV in MulBERRY multi-agent systems and **(b)** the corresponding hardware configurations.

indicates the total missions that each UAV can accomplish on a single battery charge (E), correlating to the success rate and single-mission energy as $N = SR \times E/E^{'}$. For each scenario, we evaluate 500 fault maps and report the average quantity, achieving 1.35% error margin with 95% confidence level.

It is worth mentioning that the UAV roofline model [35], simulator [2], and specifications [8, 19, 20] utilized in MulBERRY have been validated through actual UAV flights. Hence, we build upon the verified infrastructure and focus on simulation (~9000 experiments) to evaluate MulBERRY design techniques and derive insights. This approach allows for the exploration of a vast design space, which would not be possible with real UAV experiments due to non-trivial engineering and financial costs as well as safety considerations.

6.2 Hardware-Level Evaluation

Processing Platforms. As demonstrated in [51], open-source RISC-V based architectures offer a viable alternative to existing closed-source ARM-based cores in UAV platforms. Our evaluations are thus carried out on a platform consisting of 4 RISC-V Rocket cores [4] with on-chip SRAM caches. The Rocket core is placed and routed (PnR) in GF 12*nm* technology, using Cadence Genus [11] and Innovus [12] for synthesis and PnR respectively. Fig. 10 shows the post-PnR layout and the per-core configurations used in our evaluations. The per-component power is obtained at different voltage corners using the Joules Power estimator [13] on the routed netlist. The stimulus is provided in the form of a VCD file generated by running a Verilator [62] simulation of the autonomy policy model (Sec. 6.1) on the Rocket core. Along with voltage, we also scale the frequency while ensuring that the compute throughput remains higher than the sensor throughput, thus guaranteeing that the function of UAVs is unaffected (Sec. 3.1).

Sprint-or-Slack Configurations. The configurations used for sprint-or-slack are determined by evaluating the thermal limits of each UAV via HotSpot simulations [76]. For a task taking N steps, depending on the fraction of the period for which the UAV is sprinting, namely Sprinting Duty Cycle (η), we generate power traces where the UAV sprints for ηN steps and slacks for (1-η)N steps. The peak

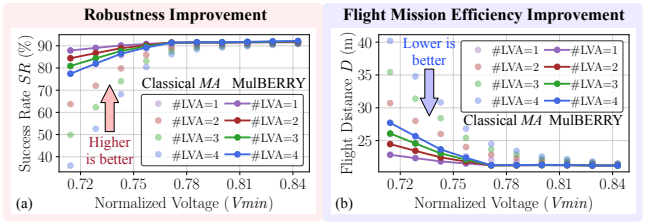


Figure 11. Robustness Improvement. Multi-UAV **(a)** average success rate and **(b)** mission completion distance under various voltages. MulBERRY improves robustness under bit failures compared to classical *MA* (Algo. 1). #LVA indicates the number of low-voltage UAVs.

temperature obtained from HotSpot across η values for the quad-Rocket core platform determines how long the UAVs can run in sprinting mode for the given heatsink configurations. η=1 means that the processor is always running at nominal voltage and frequency, likely resulting in thermal overrun. On the other hand, slacking exclusively at low voltage (η=0) results in persistent high BERs which will degrade the success rate.

7 Evaluation Results

In this section, we demonstrate that MulBERRY improves *multi-agent task robustness*, *processing-* and *mission-level efficiency* on multi-UAV systems for both offline (Sec. 7.1) and on-device robust learning scenarios (Sec. 7.2). Our evaluations are carried out for a range of environments, voltages, number and types of UAVs, autonomy policies, and chip fault patterns (Sec. 7.3).

7.1 MulBERRY Offline Robust Learning Evaluation

Robustness Improvement. Fig. 11a shows the average task success rate of multi-UAV systems under low-voltage operations with bit errors (Fig. 3). We use the classical trained multi-agent system *MA* (Algo. 1) as the baseline. Both *MA* and MulBERRY learned policies can achieve >90% average task success rate under error-free normal voltage operation. However, on lowering the voltage, the classical *MA* model is vulnerable to bit errors, while MulBERRY can still achieve 80% success rate even when all four UAVs operate at $0.73V_{min}$ or three UAVs operate at $0.71V_{min}$. The success rate usually depends on the task complexity and environmental obstacle density, and the observed mission success rate is comparable to other reported autonomous navigation tasks for similar difficulty levels [33, 36, 50].

Processing Efficiency Improvement and System Benefits. Lower operating voltage brings quadratic energy-saving benefits (Fig. 3). As in Tab. 1, compared with nominal voltage operation in [51], lowering the voltage to $0.71V_{min}$ achieves 4.05× processing energy efficiency improvement. As demonstrated on real UAVs [35], improving the processing efficiency also yields benefits for the cyber-physical components

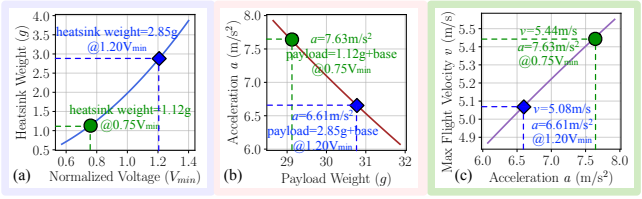


Figure 12. Benefits of Low-Voltage Operation on Cyber-physical UAV System. (a) Lowering the operating voltage results in lower energy and thermal design power (TDP), enabling a smaller heatsink with reduced weight. **(b)** Lower payload weight results in higher motion acceleration, and **(c)** Higher acceleration makes the UAV more agile when facing obstacles, enabling higher safe flight velocity. Thus, UAVs can finish a mission faster (lower flight time) with less flight energy. This enables more missions under a single battery charge.

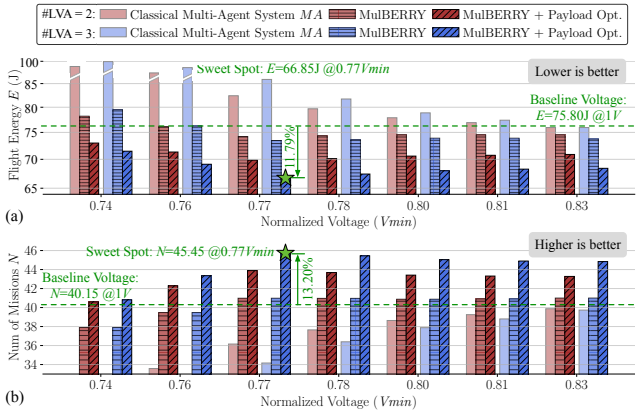


Figure 13. Mission Performance and Efficiency Improvement. MulBERRY improves multi-UAV mission efficiency with **(a)** reduced flight energy and **(b)** more number of missions owning to robust low-voltage operation. #LVA is the number of low-voltage UAVs.

(Sec. 5.3). By lowering voltage, and consequently, the Thermal Design Power (TDP), one can design a smaller heatsink with reduced weight [14] (Fig. 12a), which then yields increased motion acceleration on account of the reduced payload (Fig. 12b). With higher acceleration, the UAVs become more agile when facing obstacles and achieve higher safe flight velocity [35] (Fig. 12c). For instance, when we lower voltage from $1.20V_{min}$ to $0.75V_{min}$, the operating energy reduces 2.56× with the required heatsink weight reducing from $2.85g$ to $1.12g$, making the UAV achieve higher acceleration ($6.61m/s^2$ to $7.63m/s^2$) and safe flight velocity ($5.08m/s$ to $5.44m/s$), further enhancing autonomous system's mission-level efficiency.

System Mission Performance and Efficiency Improvement. Figs. 11b and 13 show the impact of MulBERRY on the overall mission-level performance (i.e., average task completion distance and flight energy for a single mission, and the total number of missions). We observe that the improved robustness due to MulBERRY ensures that the UAV flight

Table 1. On-Device and Offline MulBERRY Comparison. Directly learning bit errors on low-voltage chips of multi-UAVs (on-device MulBERRY) enables lower operating voltage and enhanced robustness, leading to more flight energy savings compared to offline MulBERRY, while coming at the cost of on-the-fly learning energy.

Low-Voltage Operation				Operating Efficiency	Robustness	Quality-of-Flight	
Mode	Operating Voltage	Learning Steps	Learning Energy(J)	Energy Savings	Success Rate(%)	Flight Energy(J)	Num. of Missions*
Baseline	$1V$	-	-	1×	91.4	75.80	40.15
Offline MulBERRY	$0.77V_{min}$	-	-	3.43×	91.2	66.85	45.45
	$0.71V_{min}$	-	-	4.05×	80.9	79.42	34.01
On-Device MulBERRY	$0.77V_{min}$	3000	903.0	3.43×	91.2	65.85	46.37
		5000	1505.1	3.43×	91.4	65.72	46.55
	$0.71V_{min}$	3000	872.9	4.05×	88.6	65.53	45.18
		5000	1454.8	4.05×	91.2	**64.23**	**47.62**

* Does not include on-device learning flight energy, evaluated for missions after learning.

success rate is maintained >90% and the mission completion distance ~$21m$ under $0.76V_{min}$. Notably, the maintained flight distance further reduces the single-mission flight energy. With optimized payload due to smaller heatsink at low voltage, UAVs can finish flight missions in a shorter time owning to the higher safe velocity, reducing the average flight energy from $75.80J$ to $66.85J$ (Fig. 13). The reduced flight energy and increased success rate further increase the average number of missions from 40 to 45. Interestingly, when three out of four drones operate at low-voltage (#LVA=3), MulBERRY achieves lower mission energy compared to #LVA=2 due to maintained robustness and more aggressive payload benefits. At $0.77V_{min}$, MulBERRY achieves 11.79% lower flight energy, 13.20% more missions with 3.43× operating energy savings compared to $1V$. Thus, MulBERRY enables robust and efficient low-voltage operation for multi-UAV systems.

7.2 MulBERRY On-Device Robust Learning Evaluation

On-Device and Offline MulBERRY Comparison. MulBERRY supports on-device multi-agent robust learning where UAVs can learn the bit errors directly at low-voltage chips (Sec. 5.1). While on-device learning consumes energy on the fly, compared to offline MulBERRY, UAVs can enable lower operating voltage and improved robustness due to the same fault patterns in learning and inference. The achieved lower voltage can further save more flight energy for subsequent tasks. Tab. 1 shows that with 5000 on-device learning steps, the multi-UAV system achieves robust flight under $0.71V_{min}$, resulting in 15.26% lower flight energy with 4.05× less operating energy than $1V$ operation, and 3.92% lower flight energy than offline MulBERRY. Thus, MulBERRY provides the flexibility to adapt to either offline or on-device multi-UAV robust learning depending on the scenarios.

Thermal and Sprint-or-Slack Analysis. Fig. 15 shows the peak temperature variation of the quad-core processing platform under different sprinting duty cycles (η), carried

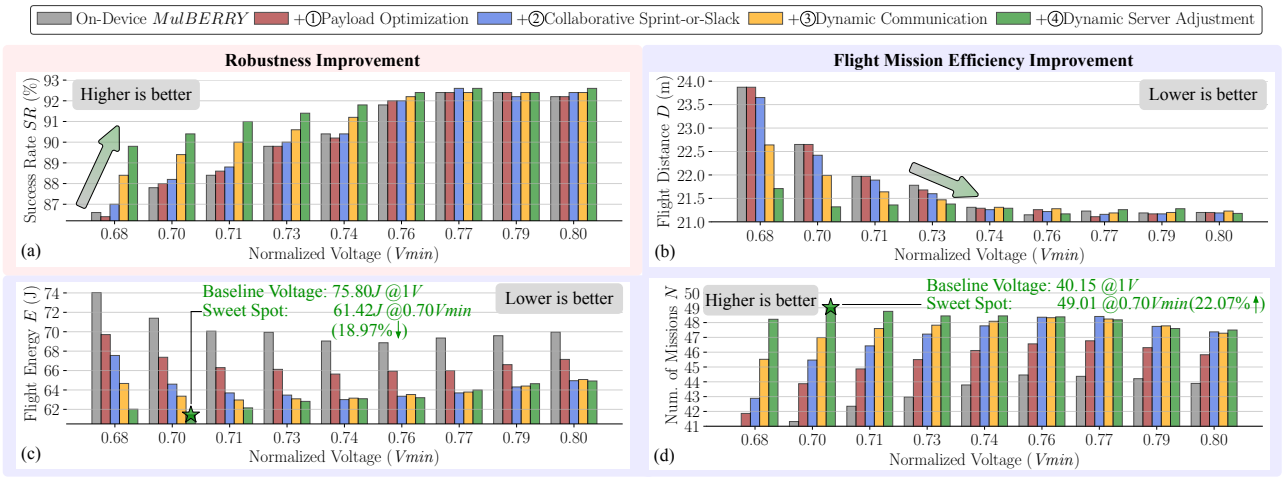


Figure 14. On-Device MulBERRY with Optimizations. Payload and collaborative sprint-or-slack optimizations are effective in improving flight agility and energy efficiency. Dynamic communication and server adjustments enhance task success rate with optimal flight distance.

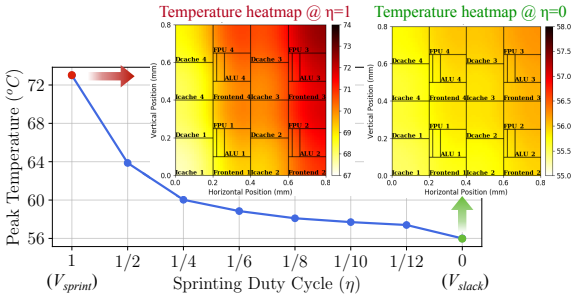
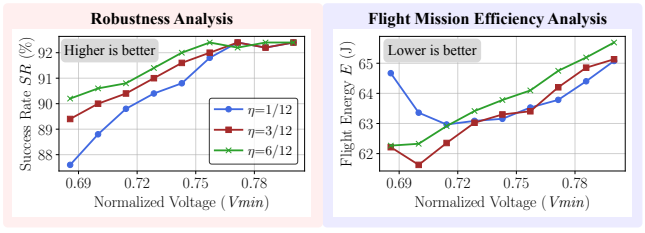


Figure 15. Thermal and Sprint-or-Slack Analysis. Peak temperature variations of the quad-core processor when the sprinting duty cycle is varied from 1 (running entirely at V_{sprint}) to 0 (running entirely at V_{slack}). The heatmaps under $\eta = 1$ and $\eta = 0$ are displayed (the heatmap scales are made differently to improve readability).



Sprinting Mode	Peak Temperature (°C)	UAV Payload (g)	Flight Velocity (m/s)
η = 1/12	57.4	29.03	2.35
η = 3/12	60.0	29.10	2.34
η = 6/12	63.9	29.23	2.32

Figure 16. Larger-Scale Multi-UAV Deployment and Sprinting Sensitivity Analysis. MulBERRY is effective in 12-UAV scenarios with robustness and mission efficiency improvements. Varying the sprinting duty cycle (η) highlights the inherent trade-off where small η degrades success rate improvement and large η is constrained by the larger heatsink, suggesting a balanced sprinting strategy.

out for an execution time of 10s with alternating sprinting and slack periods, and the heat maps for η values of 0 and 1. We observe a significant swing in the peak temperature of over 18°C (η: 1 → 0). Interestingly, our evaluation shows that the peak temperature is virtually independent of the sprinting and slack time and only dependent on η. These silicon analyses can be directly applied to multi-UAV scenarios, where one or more UAVs take turns sprinting, while the rest slack at a lower voltage. For example, when only one out of four UAVs are sprinting at any instant of time, the peak temperature reduces from ~73°C (η=1) to ~60°C (η=1/4). In other words, for a heatsink that can tolerate a peak temperature of 60°C, only a quarter of the UAVs can run at high voltage at any given time.

Robustness and Efficiency Improvements from On-Device Optimizations. Fig. 14 illustrates the multi-UAV system mission-level performance with on-device MulBERRY, enabling lower operating voltage range compared to offline MulBERRY (Fig. 13). The various optimization techniques further improve multi-UAV task robustness and efficiency. Payload optimization leads to higher safe flight velocity thus greatly reducing single mission duration and energy. The collaborative sprint-or-slack methodology enables all four UAVs to operate at low voltage (with η=1/4) without sacrificing task success rate and can improve robustness under #VLA=3 scenario, resulting in lower average mission energy across multi-UAVs. Dynamic communication and server parameter adjustments are effective in enhancing mission success rate, leading to optimal flight distance and enabling more missions. Overall, with on-device MulBERRY optimizations, at 0.70V_{min}, the multi-UAV system achieves 18.97% less flight energy, 22.07% more missions with 4.16× operating energy savings compared to 1V operation.

Larger-Scale Multi-UAV Deployment and Sprinting Sensitivity Analysis. Fig. 16 evaluates MulBERRY on a

12-UAV system and demonstrates its effective scaling to larger-scale multi-UAV systems with improved robustness and efficiency. By varying η from 1/12 (only 1 UAV sprints at a time) to 1/2 (6 UAVs sprint at a time), we observe that a higher η improves task success rate, but leads to higher peak temperature requiring a larger heatsink. This results in a higher payload and lower safe velocity. In terms of end-to-end mission efficiency, η=3/12 achieves the optimal balance with 61.62J average flight energy, whereas η=1/12 results in degraded success rate, while η=6/12 is constrained by a higher payload weight.

Communication Overhead Analysis. The communication overhead resulted from MulBERRY is determined by two main factors: the cost per communication and the total number of communications. Regarding the cost per communication, it remains unchanged from the baseline since MulBERRY does not alter the size of the policy model transmitted. While MulBERRY increases the frequency of communications for UAVs in sprinting mode, effectively doubling their communication instances, it also facilitates a reduced flight mission time due to a higher safe flight velocity (Fig. 12). This reduction in mission time compensates for the increased frequency, thus keeping the communication overhead minimal.

To put this into perspective, we quantify communication overhead for the 4-UAV, 8-UAV, and 12-UAV systems. For instance, in an 8-UAV baseline system, UAVs complete their mission in 50.2 s, involving a total of 40 agent-server communications. Under the MulBERRY framework, despite the sprinting operation doubling the frequency of communications between the UAVs and server, the UAVs finish the mission faster in only 44.6 s, resulting in the number of communications remaining unchanged at 40. This pattern is similarly observed in 4-UAV and 12-UAV systems, where the communication overheads are respectively 2.5% and 0%. This analysis shows that while MulBERRY increases communication frequency, its impact on total communication overhead is effectively mitigated by the faster mission completion times.

7.3 Sensitivity Analysis

Sensitivity across Error Patterns. In multi-UAV systems, the robust model is deployed across multiple UAVs where the chip on each UAV can have unique fault patterns. Fig. 17 evaluates MulBERRY on four profiled bit error patterns from test chips in [15, 60], one showing a random spatial error pattern and another showing a column-aligned pattern with a bias towards 0-to-1 flips. It is well observed that MulBERRY generalizes well across bit error patterns and BERs (operating voltages).

Sensitivity across Environments. Fig. 18 evaluates MulBERRY on three environments with different obstacle densities, namely sparse, medium, and dense obstacle environments. It is well observed that MulBERRY is adaptive across environments, with 19.8%, 19.0%, 16.2% single-mission flight energy reduction, and 22.3%, 22.1%,20.5% more number of missions for sparse, medium, and dense obstacle environments. and consistently improves the average flight mission efficiency. Compared across three environments, MulBERRY enables lower operating voltage in sparse environments ($0.69V_{min}$) than dense environments ($0.73V_{min}$), since a more challenging environment brings more complex trajectories for multi-UAVs to follow, which is more critical to bit errors.

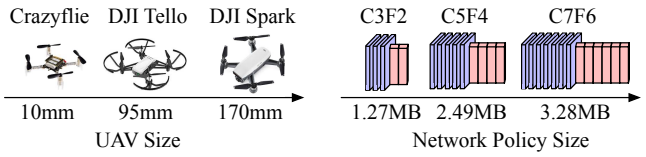
Voltage / BER (p)	Metric	UAV 1	UAV 2	UAV 3	UAV 4
Baseline $1V$ (p=0)	Success Rate = 91.4%, Flight Energy = 75.80J				
$0.77V_{min}$ / (p=0.025%)	Success Rate (%)	91.6	91.4	90.2	90.6
	Flight Energy (J)	63.90	64.06	66.16	65.47
$0.74V_{min}$ / (p=0.203%)	Success Rate (%)	91.4	91.6	90.4	90.2
	Flight Energy (J)	63.15	62.95	64.37	64.78

Figure 17. Sensitivity across Voltage and Bit Errors Patterns. MulBERRY is trained with random bit errors (p=0.2%) and generalizes well across voltage and various bit failure patterns including random and column-aligned distributions in the multi-UAV deployment, with robustness and efficiency improvements.

Environment	Sparse		Medium		Dense	
	Flight Energy (J)	Num. of Missions	Flight Energy (J)	Num. of Missions	Flight Energy (J)	Num. of Missions
Baseline @$1V$	52.41	58.56	75.80	40.15	102.4	28.04
MulBERRY (optimal)	42.02 @$0.69V_{min}$	71.63	61.42 @$0.70V_{min}$	49.01	85.77 @$0.73V_{min}$	33.79

Figure 18. Sensitivity across Environments. MulBERRY is evaluated in three environments with various obstacle densities. MulBERRY consistently improves mission efficiency (reduced flight energy and increased number of missions) and enables lower-voltage operations in sparse ($0.69V_{min}$) than complex environments ($0.73V_{min}$).

Sensitivity across UAV Types and Policies. Fig. 19 evaluates MulBERRY on two other UAV configurations (DJI Tello and Spark) in a four-UAV system. DJI Tello and Spark have larger frame sizes and takeoff weights than Crazyflie, thus the rotor power accounts for a higher ratio of total power (97.4% and 98.7%, respectively) with the same C3F2 model. Fig. 19 also evaluates MulBERRY on two other autonomy policies, C5F4 (5 Conv and 4 FC) and C7F6 (7 Conv and 6 FC), with increasing model size and compute ratio attributes compared to C3F2. Interestingly, we observe that higher compute power attributes enable MulBERRY to bring more system-level benefits. For instance, in the Crazyflie/C3F2



UAV Type	Network Policy	Rotor Power	Compute Power	MulBERRY Flight Energy ↓	MulBERRY #Missions ↑
Crazyflie	C3F2	93.5%	6.5%	18.97%	22.07%
DJI Tello	C3F2	97.4%	2.6%	13.37%	14.16%
DJI Tello	C5F4	95.0%	5.0%	16.04%	17.85%
DJI Spark	C3F2	98.7%	1.3%	6.81%	7.08%
DJI Spark	C5F4	97.5%	2.5%	12.07%	12.92%
DJI Spark	C7F6	96.7%	3.3%	13.88%	14.83%

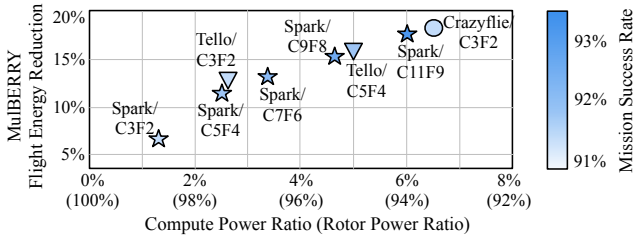


Figure 19. Sensitivity across UAVs and Models. MulBERRY is evaluated on a combination of UAVs and models, and it consistently improves task robustness and mission efficiency. A higher processing power ratio brings more system-level mission improvements. With similar processing power ratios, large UAVs can attain comparable mission energy savings to small UAVs.

configuration with a 6.5% compute power attribute, MulBERRY facilitates 18.97% mission energy savings through MulBERRY, while a DJI Spark/C3F2 configuration with a lower 1.3% compute power attribute achieves only 6.81% mission energy savings. However, when comparing UAVs with similar compute power ratios, it's noted that larger UAVs, such as those in the DJI Spark/C11F9 configuration, can attain flight energy savings comparable to smaller UAVs like the Crazyflie/C3F2 configuration. This is attributed to the ability of larger UAVs to support and operate larger models, which can lead to higher success rates due to their greater compute power and heatsink payload capacity. Hence, MulBERRY demonstrates its capability to consistently enhance task robustness and efficiency across a variety of UAVs and model configurations.

It is worth mentioning that we mainly evaluate UAVs of relatively small scales because they are typically SWaP-constrained with little-to-no offloading support. Maximizing energy efficiency in such UAVs is particularly crucial. Moreover, these smaller UAVs excel in agility and play a vital role in mainstream applications like search-and-rescue, surveillance, and the maintenance of infrastructure and equipment, as evidenced by [2, 21, 22, 36, 51].

Existing Technique Comparisons. Different low-voltage resiliency techniques target different hardware platforms and are evaluated on different scenarios, so a fair comparison is

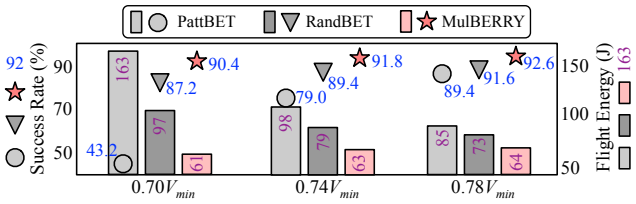


Figure 20. Existing Technique Comparisons. Compared with PattBET (trained with profiled errors) and RandBET (trained with random errors), MulBERRY generalizes across multi-UAVs and achieves improved robustness and efficiency under low voltages.

difficult. As a best effort, we compare MulBERRY with two prior techniques, where models are trained with a) profiled error patterns (PattBET) [30, 32] and b) random error patterns (RandBET) [60] (Fig. 20). PattBET, trained with fixed error patterns, does not generalize to unseen error distributions (e.g., other chips/voltages) in multi-UAVs. Moreover, profiling SRAMs requires expensive infrastructure, detailed design knowledge and time. In contrast, RandBET is designed for single-node supervised learning, and we implement and simulate it on RL-based multi-UAV systems. MulBERRY achieves 47.2% (3.2%) higher success rate, 62.5% (37%) lower energy, and 5.77× (1.64×) more number of missions at 0.70V_{min} over PattBET (RandBET) by virtual on-device robust learning and collaborative optimizations.

The energy savings of MulBERRY, especially over RandBET, are attributed to three key factors: First, MulBERRY facilitates robust operations at lower voltages through on-device learning combined with server-agent collaborative optimization. Second, it achieves higher safe flight velocities via thermal-payload optimization, leading to shorter mission durations and consequently less mission energy. Lastly, MulBERRY's higher mission success rate indicates that UAVs complete missions following optimal trajectories, which often means shorter flight distances. This integrated approach makes MulBERRY highly efficient in terms of both energy consumption and mission success.

Compatibility with Low-Power Processors or Algorithms. A benefit of MulBERRY is that the proposed robust voltage scaling and swarm learning techniques are complementary to and can be applied in conjunction with other energy-saving techniques, such as lower-power processors that are a part of commercially available UAVs, to further improve energy efficiency. For instance, we evaluate a scenario where we replace the ARM-Cortex M3 processor in a Crazyflie UAV (Cortex-UAV) with a low-power PULP processor [51] (PULP-UAV). In a representative navigation task aiming for a 90% success rate and meeting real-time requirements, the Cortex-UAV consumes 32.4 J mission energy, whereas PULP-UAV consumes 30.9 J mission energy. When MulBERRY is implemented on the PULP-UAV framework, the mission energy was further reduced by 7.1%, down to 28.7 J, while maintaining the success rate.

It is worth noting that the mere adoption of lower-power processors could potentially compromise UAVs' ability to meet real-time computational deadlines. Similarly, simpler RL algorithms might decrease the probability of successful missions. For example, [36] showed that employing simpler models with fewer layers and smaller filter sizes led to a lower overall success rate. Therefore, MulBERRY is designed to ensure that the operational efficiency of UAVs is not compromised (Fig. 4) and adopts a state-of-the-art multi-agent RL algorithm (Algo. 1), which strikes a balance between simplicity and effectiveness, meeting both accuracy and performance criteria.

8 Related Work

Robust Low-Voltage Operation. To enable robust low-voltage operation, [15, 56, 59] propose hardware-based error mitigation at the cost of power and area overhead. On the other hand, [30, 31, 60, 71] advocate improving robustness by generating model resilient to errors. Profiled errors are injected during offline training, leading to robust models during low-voltage inference. [39, 55] characterize the effect of and propose protective measures against hardware errors on DNNs of various types. However, such methods do not generalize well across voltages or patterns on multi-agent systems. MulBERRY proposes the generation of robust models but tackles completely different challenges. First, MulBERRY targets multi-agent systems, leading to a new robust framework with collaborative optimizations. Second, MulBERRY targets learning as opposed to inference, indicating that learning can occur with errors affecting parameters across agents. Third, MulBERRY tackles the cross-layer relationship of cyber-physical UAVs and brings processing and efficiency improvements.

Resilient Autonomous Systems Given the critical need for reliability in autonomous systems, several works have attempted to characterize the resilience of UAVs and other autonomous vehicles [6, 28, 29, 68]. [26, 27, 63, 64] analyze the transient error impact on the resilience of camera-based UAV autonomous navigation systems. [3, 44] characterizes the reliability of multi-UAV systems under adversarial perturbations. In contrast, MulBERRY characterizes and mitigates the effects of bit errors due to *low operating voltage* running distributed RL on multi-UAV systems with aggressive energy savings. MulBERRY supports both offline and on-device robust learning to mitigate fault impacts, achieving performance-efficiency-robustness co-optimization for autonomous systems.

Efficient Autonomous Systems. Recent efforts to develop efficient autonomous system designs include domain-specific languages [52, 57], simulation tools [9, 33, 50], benchmark suite [45], runtime [10, 58], design frameworks [34, 36, 47, 48], optimized compute units and accelerators targeting both FPGA [5, 23, 25, 41–43, 46, 66, 72] and ASICs [7, 16, 40, 51, 61, 67]. MulBERRY is complementary to and can be applied in conjunction with the above techniques, aiming to enable aggressive energy-savings yet computational-resilient autonomous systems. Furthermore, while this paper focuses on demonstrating MulBERRY in multi-UAV systems, the robust learning techniques it employs have potential applicability in a broader range of learning-based swarm systems, particularly those that are energy or resource-constrained, such as tiny robot learning [49], space and nanosatellite constellations [18].

9 Conclusion

MulBERRY is a novel multi-agent robust learning framework unlocking aggressive energy-savings yet processing-resilient swarm UAV intelligence under low-voltage operation. MulBERRY supports offline and on-device robust learning with dynamic and collaborative agent-server optimizations. Through cross-layer silicon-to-application evaluations, it achieves robustness-performance-efficiency optimizations across chips, environments, UAVs, and models, with up to 18.97% mission energy savings, 22.07% increase in successful missions with 4.16× processing energy reduction. We envision MulBERRY being useful in exploring other robust and efficient multi-agent systems.

Acknowledgements

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

References

[1] Afzal Ahmad, Viktor Walter, Pavel Petráček, Matěj Petrlík, Tomáš Báča, David Žaitlík, and Martin Saska. 2021. Autonomous aerial swarming in gnss-denied environments with high obstacle density. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 570–576.

[2] Aqeel Anwar and Arijit Raychowdhury. 2020. Autonomous navigation via deep reinforcement learning for resource constraint edge nodes using transfer learning. *IEEE Access* 8 (2020), 26549–26560.

[3] Aqeel Anwar and Arijit Raychowdhury. 2021. Multi-task federated reinforcement learning with adversaries. *arXiv preprint arXiv:2103.06473* (2021).

[4] Krste Asanovic, Rimas Avizienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, Henry Cook, Daniel Dabbelt, John Hauser, Adam Izraelevitz, et al. 2016. The rocket chip generator. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17* 4 (2016).

[5] Bahar Asgari, Ramyad Hadidi, Nima Shoghi Ghaleshahi, and Hyesoon Kim. 2020. Pisces: power-aware implementation of slam by customizing efficient sparse algebra. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[6] Subho S Banerjee, Saurabh Jha, James Cyriac, Zbigniew T Kalbarczyk, and Ravishankar K Iyer. 2018. Hands off the wheel in autonomous vehicles?: A systems perspective on over a million miles of field data. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 586–597.

[7] Pete Bannon, Ganesh Venkataramanan, Debjit Das Sarma, and Emil Talpes. 2019. Computer and redundancy solution for the full self-driving computer. In *2019 IEEE Hot Chips 31 Symposium (HCS)*. IEEE Computer Society, 1–22.

[8] Bitcraze. 2023. Crazyflie 2.1. https://www.bitcraze.io/products/crazyflie-2-1/.

[9] Behzad Boroujerdian, Hasan Genc, Srivatsan Krishnan, Wenzhi Cui, Aleksandra Faust, and Vijay Reddi. 2018. Mavbench: Micro aerial vehicle benchmarking. In *2018 51st annual IEEE/ACM international symposium on microarchitecture (MICRO)*. IEEE, 894–907.

[10] Behzad Boroujerdian, Radhika Ghosal, Jonathan Cruz, Brian Plancher, and Vijay Janapa Reddi. 2021. Roborun: A robot runtime to exploit spatial heterogeneity. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 829–834.

[11] Cadence. 2023. Genus Synthesis Solution. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html.

[12] Cadence. 2023. Innovus Implementation System. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html.

[13] Cadence. 2023. Joules RTL Power Solution. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/power-analysis/joules-rtl-power-solution.html.

[14] Celsiainc. 2023. Heat Sink Size Calculator. https://celsiainc.com/resources/calculators/heat-sink-size-calculator/.

[15] Nandhini Chandramoorthy, Karthik Swaminathan, Martin Cochet, Arun Paidimarri, Schuyler Eldridge, Rajiv V Joshi, Matthew M Ziegler, Alper Buyuktosunoglu, and Pradip Bose. 2019. Resilient low voltage accelerators for high energy efficiency. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 147–158.

[16] Muya Chang, Ashwin Sanjay Lele, Samuel D Spetalnick, Brian Crafton, Shota Konno, Zishen Wan, Ashwin Bhat, Win-San Khwa, Yu-Der Chih, Meng-Fan Chang, et al. 2023. A 73.53 TOPS/W 14.74 TOPS heterogeneous RRAM In-memory and SRAM near-memory SoC for hybrid frame and event-based target tracking. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 426–428.

[17] Timothy H. Chung (DARPA). [n. d.]. OFFensive Swarm-Enabled Tactics (OFFSET).

[18] Bradley Denby and Brandon Lucia. 2020. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 939–954.

[19] DJI. 2023. DJI Ryze Tech Tello Quadcopter. https://www.ryzerobotics.com/tello.

[20] DJI. 2023. DJI Spark Specs. https://www.dji.com/spark/info.

[21] Bardienus P Duisterhof, Srivatsan Krishnan, Jonathan J Cruz, Colby R Banbury, William Fu, Aleksandra Faust, Guido CHE de Croon, and Vijay Janapa Reddi. 2021. Tiny robot learning (tinyrl) for source seeking on a nano quadcopter. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 7242–7248.

[22] Bardienus P Duisterhof, Shushuai Li, Javier Burgués, Vijay Janapa Reddi, and Guido CHE de Croon. 2021. 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)*. IEEE, 9099–9106.

[23] Yiming Gan, Yu Bo, Boyuan Tian, Leimeng Xu, Wei Hu, Shaoshan Liu, Qiang Liu, Yanjun Zhang, Jie Tang, and Yuhao Zhu. 2021. Eudoxus: Characterizing and accelerating localization in autonomous machines industry track paper. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 827–840.

[24] Shrikanth Ganapathy, John Kalamatianos, Keith Kasprak, and Steven Raasch. 2017. On characterizing near-threshold SRAM failures in FinFET technology. In *Proceedings of the 54th Annual Design Automation Conference (DAC) 2017*. 1–6.

[25] Tian Gao, Zishen Wan, Yuyang Zhang, Bo Yu, Yanjun Zhang, Shaoshan Liu, and Arijit Raychowdhury. 2021. IELAS: An ELAS-based energy-efficient accelerator for real-time stereo matching on FPGA platform. In *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 1–4.

[26] Yu-Shun Hsiao, Zishen Wan, Tianyu Jia, Radhika Ghosal, Abdulrahman Mahmoud, Arijit Raychowdhury, David Brooks, Gu-Yeon Wei, and Vijay Janapa Reddi. 2023. 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)*. IEEE, 1–6.

[27] Yu-Shun Hsiao, Zishen Wan, Tianyu Jia, Radhika Ghosal, Abdulrahman Mahmoud, Arijit Raychowdhury, David Brooks, Gu-Yeon Wei, and Vijay Janapa Reddi. 2023. Silent Data Corruption in Robot Operating System: A Case for End-to-End System-Level Fault Analysis Using Autonomous UAVs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2023).

[28] Saurabh Jha, Shengkun Cui, Timothy Tsai, Siva Kumar Sastry Hari, Michael B Sullivan, Zbigniew T Kalbarczyk, Stephen W Keckler, and Ravishankar K Iyer. 2022. Exploiting temporal data diversity for detecting safety-critical faults in AV compute systems. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 88–100.

[29] Saurabh Jha, Timothy Tsai, Siva Hari, Michael Sullivan, Zbigniew Kalbarczyk, Stephen Keckler, and Ravishankar Iyer. 2018. Kayotee: A Fault Injection-based System to Assess the Safety and Reliability of Autonomous Vehicles to Faults and Errors. In *IEEE International Workshop on Automotive Reliability Test.*

[30] Sung Kim, Patrick Howe, Thierry Moreau, Armin Alaghi, Luis Ceze, and Visvesh Sathe. 2018. MATIC: Learning around errors for efficient low-voltage neural network accelerators. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1–6.

[31] Skanda Koppula, Lois Orosa, A Giray Yağlıkçı, Roknoddin Azizi, Taha Shahroodi, Konstantinos Kanellopoulos, and Onur Mutlu. 2019. 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)*. 166–181.

[32] Skanda Koppula, Lois Orosa, A Giray Yağlıkçı, Roknoddin Azizi, Taha Shahroodi, Konstantinos Kanellopoulos, and Onur Mutlu. 2019. 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)*. 166–181.

[33] Srivatsan Krishnan, Behzad Boroujerdian, William Fu, Aleksandra Faust, and Vijay Janapa Reddi. 2021. Air learning: a deep reinforcement learning gym for autonomous aerial robot visual navigation. *Machine Learning* 110 (2021), 2501–2540.

[34] Srivatsan Krishnan, Thierry Tambe, Zishen Wan, and Vijay Janapa Reddi. 2021. Autosoc: Automating algorithm-soc co-design for aerial robots. *arXiv preprint arXiv:2109.05683* (2021).

[35] Srivatsan Krishnan, Zishen Wan, Kshitij Bhardwaj, Ninad Jadhav, Aleksandra Faust, and Vijay Janapa Reddi. 2022. 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)*. IEEE, 162–174.

[36] Srivatsan Krishnan, Zishen Wan, Kshitij Bhardwaj, Paul Whatmough, Aleksandra Faust, Sabrina Neuman, Gu-Yeon Wei, David Brooks, and Vijay Janapa Reddi. 2022. Automatic Domain-Specific SoC Design

for Autonomous Unmanned Aerial Vehicles. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 300–317.

[37] Srivatsan Krishnan, Zishen Wan, Kshitij Bhardwaj, Paul Whatmough, Aleksandra Faust, Gu-Yeon Wei, David Brooks, and Vijay Janapa Reddi. 2020. The sky is not the limit: A visual performance model for cyber-physical co-design in autonomous machines. *IEEE Computer Architecture Letters* 19, 1 (2020), 38–42.

[38] Animesh Kumar, Jan Rabaey, and Kannan Ramchandran. 2009. SRAM supply voltage scaling: A reliability perspective. In *2009 10th international symposium on quality electronic design*. IEEE, 782–787.

[39] Guanpeng Li, Siva Kumar Sastry Hari, Michael Sullivan, Timothy Tsai, Karthik Pattabiraman, Joel Emer, and Stephen W Keckler. 2017. Understanding error propagation in deep learning neural network (DNN) accelerators and applications. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. 1–12.

[40] Ziyun Li, Yu Chen, Luyao Gong, Lu Liu, Dennis Sylvester, David Blaauw, and Hun-Seok Kim. 2019. An 879gops 243mw 80fps vga fully visual cnn-slam processor for wide-range autonomous exploration. In *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 134–136.

[41] Qiang Liu, Zishen Wan, Bo Yu, Weizhuang Liu, Shaoshan Liu, and Arijit Raychowdhury. 2022. An energy-efficient and runtime-reconfigurable fpga-based accelerator for robotic localization systems. In *2022 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 01–02.

[42] Shaoshan Liu, Zishen Wan, Bo Yu, and Yu Wang. 2021. *Robotic computing on fpgas*. Springer.

[43] Weizhuang Liu, Bo Yu, Yiming Gan, Qiang Liu, Jie Tang, Shaoshan Liu, and Yuhao Zhu. 2021. Archytas: A framework for synthesizing and dynamically optimizing accelerators for robotic localization. In *2021 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 479–493.

[44] Chuan Ma, Jun Li, Kang Wei, Bo Liu, Ming Ding, Long Yuan, Zhu Han, and H Vincent Poor. 2022. Trusted AI in multi-agent systems: An overview of privacy and security for distributed learning. *arXiv preprint arXiv:2202.09027* (2022).

[45] Víctor Mayoral-Vilches, Jason Jabbour, Yu-Shun Hsiao, Zishen Wan, Alejandra Martínez-Fariña, Martino Crespo-Alvarez, Matthew Stewart, Juan Manuel Reina-Munoz, Prateek Nagras, Gaurav Vikhe, et al. 2023. RobotPerf: An Open-Source, Vendor-Agnostic, Benchmarking Suite for Evaluating Robotics Computing System Performance. *arXiv preprint arXiv:2309.09212* (2023).

[46] Sean Murray, William Floyd-Jones, Ying Qi, George Konidaris, and Daniel J Sorin. 2016. The microarchitecture of a real-time robot motion planning accelerator. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 1–12.

[47] Sabrina M Neuman, Radhika Ghosal, Thomas Bourgeat, Brian Plancher, and Vijay Janapa Reddi. 2023. RoboShape: Using Topology Patterns to Scalably and Flexibly Deploy Accelerators Across Robots. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA)*. 1–13.

[48] Sabrina M Neuman, Brian Plancher, Thomas Bourgeat, Thierry Tambe, Srinivas Devadas, and Vijay Janapa Reddi. 2021. Robomorphic computing: a design methodology for domain-specific accelerators parameterized by robot morphology. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 674–686.

[49] Sabrina M Neuman, Brian Plancher, Bardienus P Duisterhof, Srivatsan Krishnan, Colby Banbury, Mark Mazumder, Shvetank Prakash, Jason Jabbour, Aleksandra Faust, Guido CHE de Croon, et al. 2022. Tiny robot learning: challenges and directions for machine learning in resource-constrained robots. In *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 296–299.

[50] Dima Nikiforov, Shengjun Chris Dong, Chengyi Lux Zhang, Seah Kim, Borivoje Nikolic, and Yakun Sophia Shao. 2023. RoSÉ: A Hardware-Software Co-Simulation Infrastructure Enabling Pre-Silicon Full-Stack Robotics SoC Evaluation. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA)*. 1–15.

[51] Daniele Palossi, Antonio Loquercio, Francesco Conti, Eric Flamand, Davide Scaramuzza, and Luca Benini. 2019. A 64-mw dnn-based visual navigation engine for autonomous nano-drones. *IEEE Internet of Things Journal* 6, 5 (2019), 8357–8371.

[52] Liam Patterson, David Pigorovsky, Brian Dempsey, Nikita Lazarev, Aditya Shah, Clara Steinhoff, Ariana Bruno, Justin Hu, and Christina Delimitrou. 2022. HiveMind: a hardware-software system stack for serverless edge swarms. In *Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA)*. 800–816.

[53] Arun Raghavan, Laurel Emurian, Lei Shao, Marios Papaefthymiou, Kevin P Pipe, Thomas F Wenisch, and Milo MK Martin. 2013. Computational sprinting on a hardware/software testbed. *ACM SIGARCH Computer Architecture News* 41, 1 (2013), 155–166.

[54] Arun Raghavan, Yixin Luo, Anuj Chandawalla, Marios Papaefthymiou, Kevin P Pipe, Thomas F Wenisch, and Milo MK Martin. 2012. Computational sprinting. In *IEEE international symposium on high-performance comp architecture (HPCA)*. IEEE, 1–12.

[55] Brandon Reagen, Udit Gupta, Lillian Pentecost, Paul Whatmough, Sae Kyu Lee, Niamh Mulholland, David Brooks, and Gu-Yeon Wei. 2018. Ares: A framework for quantifying the resilience of deep neural networks. In *Proceedings of the 55th Annual Design Automation Conference (DAC)*. 1–6.

[56] Brandon Reagen, Paul Whatmough, Robert Adolf, Saketh Rama, Hyunkwang Lee, Sae Kyu Lee, José Miguel Hernández-Lobato, Gu-Yeon Wei, and David Brooks. 2016. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. In *ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 267–278.

[57] Jacob Sacks, Divya Mahajan, Richard C Lawson, Behnam Khaleghi, and Hadi Esmaeilzadeh. 2018. Robox: an end-to-end solution to accelerate autonomous control in robotics. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 479–490.

[58] Deval Shah, Ningfeng Yang, and Tor M Aamodt. 2023. Energy-Efficient Realtime Motion Planning. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA)*. 1–17.

[59] Gopalakrishnan Srinivasan, Parami Wijesinghe, Syed Shakib Sarwar, Akhilesh Jaiswal, and Kaushik Roy. 2016. Significance driven hybrid 8T-6T SRAM for energy-efficient synaptic storage in artificial neural networks. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 151–156.

[60] David Stutz, Nandhini Chandramoorthy, Matthias Hein, and Bernt Schiele. 2021. Bit error robustness for energy-efficient dnn accelerators. *Proceedings of Machine Learning and Systems (MLSys)* 3 (2021), 569–598.

[61] Amr Suleiman, Zhengdong Zhang, Luca Carlone, Sertac Karaman, and Vivienne Sze. 2019. Navion: A 2-mw fully integrated real-time visual-inertial odometry accelerator for autonomous navigation of nano drones. *IEEE Journal of Solid-State Circuits* 54, 4 (2019), 1106–1119.

[62] Verilator. 2023. Verilator. https://www.veripool.org/verilator.

[63] Zishen Wan, Aqeel Anwar, Yu-Shun Hsiao, Tianyu Jia, Vijay Janapa Reddi, and Arijit Raychowdhury. 2021. Analyzing and improving fault tolerance of learning-based navigation systems. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 841–846.

[64] Zishen Wan, Aqeel Anwar, Abdulrahman Mahmoud, Tianyu Jia, Yu-Shun Hsiao, Vijay Janapa Reddi, and Arijit Raychowdhury. 2022. Frl-fi: Transient fault analysis for federated reinforcement learning-based

navigation systems. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 430–435.

[65] Zishen Wan, Nandhini Chandramoorthy, Karthik Swaminathan, Pin-Yu Chen, Vijay Janapa Reddi, and Arijit Raychowdhury. 2023. BERRY: Bit Error Robustness for Energy-Efficient Reinforcement Learning-Based Autonomous Systems. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[66] Zishen Wan, Ashwin Lele, Bo Yu, Shaoshan Liu, Yu Wang, Vijay Janapa Reddi, Cong Hao, and Arijit Raychowdhury. 2022. Robotic computing on fpgas: Current progress, research challenges, and opportunities. In *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 291–295.

[67] Zishen Wan, Ashwin Sanjay Lele, and Arijit Raychowdhury. 2022. Circuit and system technologies for energy-efficient edge robotics. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 275–280.

[68] Zishen Wan, Karthik Swaminathan, Pin-Yu Chen, Nandhini Chandramoorthy, and Arijit Raychowdhury. 2022. Analyzing and Improving Resilience and Robustness of Autonomous Systems. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–9.

[69] Zishen Wan, Bo Yu, Thomas Yuang Li, Jie Tang, Yuhao Zhu, Yu Wang, Arijit Raychowdhury, and Shaoshan Liu. 2021. A survey of fpga-based robotic computing. *IEEE Circuits and Systems Magazine* 21, 2 (2021), 48–74.

[70] Jingao Xu, Hao Cao, Zheng Yang, Longfei Shangguan, Jialin Zhang, Xiaowu He, and Yunhao Liu. 2022. SwarmMap: Scaling up real-time collaborative visual SLAM at the edge. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 977–993.

[71] Lita Yang and Boris Murmann. 2017. SRAM voltage scaling for energy-efficient convolutional neural networks. In *2017 18th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 7–12.

[72] Bo Yu, Wei Hu, Leimeng Xu, Jie Tang, Shaoshan Liu, and Yuhao Zhu. 2020. Building the computing system for autonomous micromobility vehicles: Design constraints and architectural optimizations. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 1067–1081.

[73] Sihan Zeng, Malik Aqeel Anwar, Thinh T Doan, Arijit Raychowdhury, and Justin Romberg. 2021. A decentralized policy gradient approach to multi-task reinforcement learning. In *Uncertainty in Artificial Intelligence (UAI)*. PMLR, 1002–1012.

[74] Jeff Zhang, Kartheek Rangineni, Zahra Ghodsi, and Siddharth Garg. 2018. Thundervolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators. In *Proceedings of the 55th Annual Design Automation Conference (DAC)*. 1–6.

[75] Jeff Jun Zhang, Kang Liu, Faiq Khalid, Muhammad Abdullah Hanif, Semeen Rehman, Theocharis Theocharides, Alessandro Artussi, Muhammad Shafique, and Siddharth Garg. 2019. Building robust machine learning systems: Current progress, research challenges, and opportunities. In *Proceedings of the 56th Annual Design Automation Conference (DAC)*. 1–4.

[76] Runjie Zhang, Mircea R Stan, and Kevin Skadron. 2015. Hotspot 6.0: Validation, acceleration and extension. *University of Virginia, Tech. Rep* (2015).