Analyzing and Improving Resilience and Robustness of Autonomous Systems

(Invited Paper)

Zishen Wan[1], Karthik Swaminathan[2], Pin-Yu Chen[2], Nandhini Chandramoorthy[2]
Arijit Raychowdhury[1]
[1]Georgia Institute of Technology, Atlanta, GA [2]IBM T.J. Watson Research Center, Yorktown Heights, NY

ABSTRACT

Autonomous systems have reached a tipping point, with a myriad of self-driving cars, unmanned aerial vehicles (UAVs), and robots being widely applied and revolutionizing new applications. The continuous deployment of autonomous systems reveals the need for designs that facilitate increased resiliency and safety. The ability of an autonomous system to tolerate, or mitigate against errors, such as environmental conditions, sensor, hardware and software faults, and adversarial attacks, is essential to ensure its functional safety. Application-aware resilience metrics, holistic fault analysis frameworks, and lightweight fault mitigation techniques are being proposed for accurate and effective resilience and robustness assessment and improvement. This paper explores the origination of fault sources across the computing stack of autonomous systems, discusses the various fault impacts and fault mitigation techniques of different scales of autonomous systems, and concludes with challenges and opportunities for assessing and building next-generation resilient and robust autonomous systems.

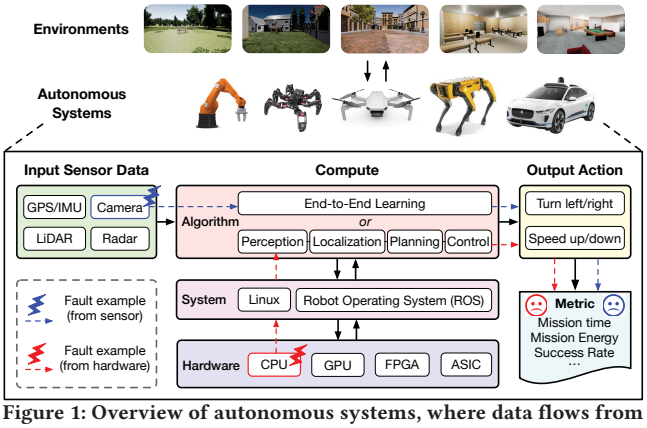


Figure 1: Overview of autonomous systems, where data flows from environment to sensors, going through compute (with hardware-system-algorithm support) to output actions in a loop. Faults in any stage may propagate through the cross-layer closed-loop autonomous system and impact the final performance and reliability.

1 INTRODUCTION

The advent of autonomous systems, including self-driving cars, Unmanned Aerial Vehicles (UAVs), and robots, has completely revolutionalized several application domains such as automotive, aviation, and agriculture, to name a few [1–3]. The overall market for autonomous vehicles is expected to grow 30-fold over this decade to over $2 Trillion in 2030 [4]. This has further spurred a series of innovations both at the algorithm and system-level, resulting in large-scale deployment such as *Intelligent Swarms*, increased autonomy, and highly efficient custom hardware designs at low form factors [5–10].

Like any computing system at-scale, autonomous systems also face several challenges in deployment, primarily the need for designs that facilitate increased resilience and safety. Increased transistor density and complexity in the overall manufacturing process that arises out of transistor scaling to sub-10 nm nodes only serve to exacerbate these challenges. Autonomous systems, particularly those operating in harsh environmental conditions, can encounter various sources of errors and potential failures. These include radiation-induced soft or transient errors (SER), timing errors or memory bit failures due to voltage over-scaling, aging or on-field permanent failures, changes in the input data due to malfunctioning sensors or environmental conditions, or even Silent Data Corruptions (SDCs) that were recently discovered at-scale resulting in mercurial or unpredictable functionality in certain cores or computing units [11]. Further, optimizations targeted towards maximizing the performance and power efficiency, such as operating at reduced supply voltages, lower precision operation, or lower resolution inputs, can have adverse consequences on the overall reliability. Finally, targeted or adversarial errors, possibly from malicious entities, can have catastrophic effects on the functionality of autonomous systems. A small error in the model parameters or adversarial or data poisoning attacks on the inputs can result in a significant increase in misclassifications.

The ability of an autonomous system to tolerate or mitigate these errors is essential to ensure its functional safety. Recent industry studies [12, 13] have shown that evaluating the resilience of individual components or parts of the compute/control stack is devoid of a full end-to-end and cross-stack perspective that may both limit reliability solutions as well as miss error propagation altogether through a complex system. In contrast to standard metrics such as *Mean-Time-To-Failure (MTTF)* used for large-scale cloud and datacenter systems, autonomous system resilience highly depends on the application, sensor inputs, and the nature of the environment. As a result existing methodologies for evaluating and ensuring Reliability, Availability, and Serviceability (RAS) in CPU and GPU-based multiprocessors, for instance, would not be effective in this context. This underscores the need to develop new metrics, error detection, and mitigation strategies specifically targeted for such systems. For instance, estimating the end-to-end reliability of an autonomous system that comprises an aggregation of several general-purpose and custom computing components continues to remain a challenge even today.

ICCAD '22, October 30-November 3, 2022, San Diego, CA, USA

ACM ISBN 978-1-4503-9217-4/22/10.
https://doi.org/10.1145/3508352.3561111

Table 1: Exampled fault sources in autonomous systems.

Resilience	**Input Data Fault**	Environmental conditions (e.g., blur, contrast, brightness) Sensor noise (e.g., camera, IMU, LiDAR, GPS) Unreliable or tainted information from swarm agents
	Hardware Fault	Soft errors (radiation) Memory bit failures (voltage over-scaling) Timing errors (voltage droops, overclocking) Aging/on-field permanent failures
	Software Fault	Software bugs Incorrect or overly aggressive implementation
Robustness	**Adversarial Attack**	Adversarial attack on ML model (perception, E2E learning) Data poisoning attack on input data
	Hardware Error	Targeted hardware errors (Rowhammer, corrupted model)

In this paper, we explore the key considerations involved in designing autonomous systems that are highly efficient, but are also resilient to faults across the computing stack. We highlight key observations on the types of faults that are of interest in this domain, metrics that determine the impact of these faults on the overall system functionality, and methodologies for modeling, detecting, and mitigating the errors induced due to these faults. Finally, we present a series of challenges and opportunities for the research community that could be instrumental in the development of reliable, efficient, and scalable autonomous systems of the future.

2 AUTONOMOUS SYSTEMS AND FAULT SOURCES

This section introduces the autonomous system and its fault sources. We first present an overview of the closed-loop cross-layer autonomous system computing stack (Section 2.1). Then we highlight the various fault originations that will impact the resilience and robustness of autonomous systems (Section 2.2).

2.1 Autonomous Systems

Autonomous systems typically operate in a closed-loop manner, where the data flows from the environment, going through the autonomous system and back to the environment, as shown in Fig. 1. This procedure involves sensing the environment (input data), making autonomous decisions (compute), and finally actuating within the environment (output action) in a loop.

Cross-layer compute stack is an integral component of the closed-loop autonomous systems, spanning from autonomy algorithm, system, and compute hardware, to achieve intelligence. Autonomy algorithms interpret the environment and make decisions, which typically includes two paradigms, namely physical model-based autonomy (e.g., perception, localization, planning, control) and learning-based autonomy (e.g., end-to-end learning). The system layer provides communication functions and resource allocation for autonomous applications with ROS [14], maps workloads to compute units, and schedules tasks at runtime with Linux. Compute hardware then executes algorithm kernels with the support of different substrate platforms (e.g., CPU, GPU, FPGA, ASIC). All of these cross-layer components work as a coherent closed-loop system to achieve autonomous intelligence.

2.2 Fault Sources

Understanding where a fault originates from is necessary for analyzing how it propagates through the autonomous system to impact resiliency and safety. We identify different fault sources and broadly categorize them based on two characteristics: resilience and robustness, as summarized in Table 1.

Table 2: Comparison of reliability evaluation metrics between processors and autonomous systems.

Conventional Reliability Metric	Application-aware Reliability Metric	
Failure-in-Time (FIT) rate Mean-time-between-failure (MTBF)	UAV [23]	Flight distance Flight energy Navigation success rate
	Car [22, 24]	Mission success rate #Traffic violations #Accidents Time to traffic violation Stopping distance
	Robot [25]	Collision Exposure Factor

2.2.1 Resilience. Resilience is the ability of autonomous systems to tolerate errors that *randomly occur* occurred at input data and various levels of the computing organizations.

Input data fault: Faults in input data can arise from environmental conditions, sensors, and other agents in swarm systems. Various occlusion, contrast, brightness, and blur conditions from environments may add perturbations in sensing images and degrade perception accuracy. Noise and malfunction of sensors (e.g., camera, IMU, LiDAR, GPS) may alter the input information and modify the autonomous systems' interpretation of surrounding states. Unreliable or tainted information from other agents in collaborative systems may confuse normal agents and result in wrong decisions.

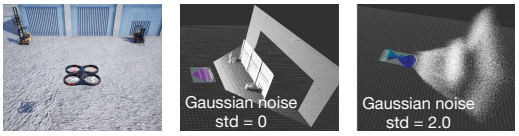
Hardware fault: Hardware faults include radiation-induced soft errors [15], memory bit failures due to voltage over-scaling [16], timing errors due to voltage droops or overclocking [17], and aging or on-field permanent stuck-at failures that are repeatable and occur the same way every time [18]. For example, a recent chip characterization study reveals that lowering operation voltage brings bit-flips in on-chip SRAM cells, and the failures consistently exist under voltage scaling [19]. These failures may impact both memory and compute units and exacerbate with the continuously increased technology node density, wider datapaths, and voltage scaling.

Software fault: Software faults originate from either software bugs in computer software or overly aggressive implementation, such as software approximation and low precision [20, 21]. Timing faults in network communication paths (e.g., data loss, out-of-order or delayed data delivery) [22] may result in unexpected and incorrect values in the autonomous system as well.

2.2.2 Robustness. Robustness is the ability of autonomous systems to tolerate errors that are *maliciously induced*, such as adversarial attacks and targeted hardware errors. Adversarial attacks particularly impact input data or ML-based models. The poisoned data may make the sensing images suspicious and be maliciously tampered with to mislead output action. The attacked ML model (e.g., in perception or end-to-end learning) may intend to generate wrong decisions with the correct information or seek to increase compute latency and energy consumption of autonomous systems. Targeted hardware errors, such as Rowhammer [26], ClkScrew attack [27] and hardware fault attack [28] may result in contaminated storage data and wrong compute results, posing threats to the autonomous systems. The distributed on-chip power control mechanism could be subject to malicious or inadvertent energy attacks [29].

Brightness	Contrast	Flight Time (s)	Flight Energy (kJ)	Success Rate (%)
No noise (b=0)	No noise (c=1)	124.3	67.5	100
b = 45	No noise (c=1)	131.6	72.2	99
b = 90	No noise (c=1)	145.1	77.9	96
No noise (b=0)	c = 1.5	139.3	75.1	97
No noise (b=0)	c = 2.0	162.7	89.8	92

Figure 2: The impact of environmental conditions (e.g., brightness and contrast) on UAV autonomous system performance and safety.



Gaussian Noise std (m)	Flight Time (s)	Flight Energy (kJ)	Success Rate (%)
0	124.3	67.5	100
0.5	130.1	69.1	99
1.0	155.4	86.9	95
1.5	189.6	108.2	90
2.0	261.7	144.8	82

Figure 3: The impact of senor noises (e.g., Gaussian noise in camera depth reading) on UAV autonomous system performance and safety.

3 FAULT IMPACT ON AUTONOMOUS SYSTEMS

This section studies the fault impact on autonomous systems. First, we discuss application-aware metrics for accurate and effective resilience evaluation (Section 3.1). Next, we examine various fault impacts on autonomous systems with an example navigation task (Section 3.2). Finally, we review the existing fault injection framework and highlight the significance of intelligent fault injection for complex autonomous systems (Section 3.3).

3.1 Metric

Observation 1: *Identifying application-aware metrics is critical for holistic and effective autonomous system resilience evaluation.*

Identifying correct metrics and establishing safety violation conditions paves the way for evaluating autonomous system resilience. Conventional reliability metrics, such as failure-in-time (FIT) rate and mean-time-between-failure (MTBF), are effective in evaluating individual component resilience (e.g., CPU and GPU processor). However, these metrics miss capturing the fault propagation through various components in autonomous systems.

Application-level performance and safety characteristics should be incorporated into resilience metrics to reveal fault impact. Table 2 presents exampled application-aware reliability metrics for three autonomous applications. For UAVs, mission flight time, flight energy, and success rate of the autonomous navigation task can be used for UAV system resilience evaluation [23]. For self-driving cars, AVFI [22] quantifies mission success rate, the number of traffic violations, and time to traffic violation as application-aware failure metrics to evaluate safety. Similarly, DriveFI [24] defines the instantaneous safety criteria and stopping distance based on collision avoidance conditions. For robots, Shah et al. [25] define collision exposure factor as a metric to assess the failure circuit vulnerability of motion planning task, based on the relation between physical space and safety violation cases. These application-aware metrics are able to capture end-to-end fault propagation across computing stack, and provide guidance in intelligent fault injection and designing efficient domain-specific resilient techniques.

3.2 Fault Impact

Fault originated at various places (Section 2.2) may propagate through the autonomous system and impact the application performance and reliability (Fig. 1). Using application-aware metrics (Section 3.1), we examine the fault impact on autonomous systems.

Observation 2: *Environmental conditions and sensor noises may degrade the surrounding interpretation, and impact the performance and safety of autonomous systems.*

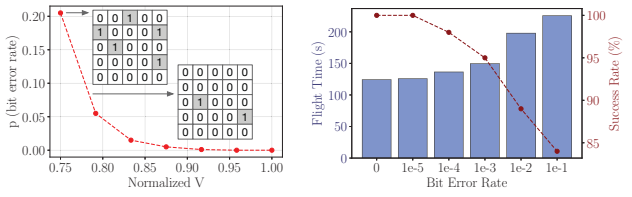
Fig. 2 demonstrates the environmental impact on autonomous navigation applications, where we investigate two types of conditions: brightness and contrast. We use UAV autonomous navigation as a case study, where the task of the UAV is to fly from the start position to the goal position in the shortest time without colliding into any obstacles. The experiments are based on UAV simulators MAVBench [30] and PEDRA [31]. For environment noise, we apply noise model $f(I) = bI + c$ on sensing images, where I represents images, b and c represent brightness and contrast factor, respectively. We evaluate $\{b = 0, 45, 90\}$ and $\{c = 1.0, 1.5, 2.0\}$ cases, and observe that increased brightness and contrast result in lower task success rate with higher mission time and energy. This is because brightness and contrast make some objects appear whiter and increase the difficulty for UAVs to detect, resulting in path detours and more path planning actions. Various real-world weather conditions, such as rain, fog, and snow, can manifest as different types of fault in sensing images and result in potential safety violations [32].

Fig. 3 assesses the sensor noise impact on UAV autonomous navigation application, where we inject Gaussian noise in depth reading of RGBD camera with various scales. We observe that Gaussian noise inflates and obscures the objects, making the UAV plan the trajectory more often since the original planned path may be falsely perceived to lead to a collision. This causes the average flight time to increase by 2.1×, with 2.1× more energy consumption. False perception can even fail the navigation task, either making the UAV collide into obstacles or fail to find feasible paths, resulting in up to 18% success rate drop. This observation aligns with [30].

Faults from other sensor modules can impact application safety as well. For example, [32] reveals that Radar faults will degrade distance and velocity information, and propagate into self-driving car systems, resulting in unsafe and traffic rule violations in both high-speed and low-speed driving scenarios. Reproducible and rigorous fault models for each type of sensor need to be established to facilitate accurate end-to-end resilience assessment.

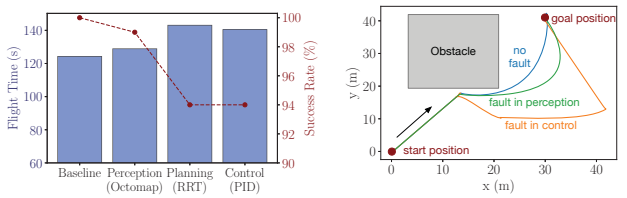
Observation 3: *Hardware faults in memory and compute units can propagate through the autonomous system and impact application performance and safety. Meanwhile, the frontend module (e.g., perception) demonstrates higher resilience than the backend module (e.g., planning and control).*

Fig. 4 shows the memory bit failure impact on a learning-based autonomous navigation task. Recent chip characterization studies [16, 33] reveal that aggressive SRAM supply voltage scaling for



(a) Memory bit failure model [16]. (b) Memory bit failure impact.

Figure 4: The impact of memory bit failures (due to voltage over-scaling) on UAV autonomous system performance and safety.



(a) Compute SDC impact. (b) Exampled trajectories under SDC.

Figure 5: The impact of compute silent data corruption (SDC) on UAV autonomous system performance and safety.

energy efficiency improvement will cause bit-level failures in SRAM on account of process variation. The error rate increases exponentially with lowered voltage, and errors are persistent across supply voltages for a fixed memory array (Fig. 4(a)). Applying the memory fault model to a learning-based autonomous navigation task, we observe that learning-based autonomy has inherent resilience when the number of bit-flips is small. However, as the bit error rate increases, the errors have higher chances of propagating through the system and making UAV detour optimal trajectory, resulting in longer flight time or even collision to fail the task (Fig. 4(b)).

Fig. 5 explores the silent data corruption impact on a physical model-based autonomous navigation task. Physical model-based autonomy typically involves multiple stages, where the perception stage builds a detailed representation of surroundings and locates the agent, the planning stage finds an optimal collision-free path, followed by the control stage to continuously track the differences between actual poses and pre-planned path and move the agent. We pick a typical kernel in each stage and inject a single transient bit-flip at the source or destination register of execution instruction [23]. The fault emulates silent data corruption in the processor's functional units. As Fig. 5(a) shows, we observe that in this application, fault in fronted perception has less impact on task performance, while faults in backend planning and control have a higher chance of leading to task failure. This is because perception has more information redundancy than planning and control. One corrupted voxel in perception may still be remedied by other normal voxels and result in correct planning results, however, backend stages usually only include critical information and can directly lead to a detour (Fig. 5(b)) [23]. Similarly, in a self-driving car, fault in the backend compute module (e.g., throttle, PID controller, steer) brings more traffic violation [24]. This observation provides opportunities for intelligent and adaptive fault mitigation.

Observation 4: *Adversarial attacks in data collection, model training, and model deployment pose reliability and safety threats to autonomous systems.*

Adversarial robustness aims to study the weaknesses of a model in its lifecycle that could possibly be exploited by a bad actor and result in negative impacts [34]. The lifecycle spans data collection and processing, model training, and model deployment. A *threat model* specifies the capability of the bad actor and the attacker's objective. For example, *data poisoning* attack assumes an attacker can manipulate a subset of the training data such that models trained on the tampered dataset will carry certain vulnerabilities to be exploited, such as a backdoor attack that is able to control the prediction of the model upon the presence of some trigger patter at the data input. Another well-known weakness is *adversarial example*, a crafted perturbed data sample to evade the prediction of a target model while maintaining high similarity to the original natural instances. These adversarial examples can also be realized in the physical space as adversarial objects. In particular, many physical adversarial examples are shown to be evasive to object detectors (e.g., adversarial stop sign [35], adversarial T-Shirt [36], and adversarial make-up [37]), which is an essential component in autonomous systems. For instance, the authors in [36] showed that their adversarial T-Shirt could achieve 74% and 57% attack success rates in the digital and physical worlds against YOLOv2, respectively, which means in a majority of the frames, the target model makes incorrect object detections for a person wearing the adversarial T-Shirt. There are also specialized adversarial attacks proposed for autonomous systems, such as the phantom attack targeting advanced driver-assistance systems [38]. Many attacks can be executed given limited information and feedback, such as using the principle of query-based black-box attacks [39, 40].

3.3 Fault Injection Tool and Methodology

Observation 5: *Intelligent fault injection (FI) scheme can reduce FI experiment time, prune FI test space, and efficiently identify safety-critical scenarios.*

Fault injection is a critical and well-established technique for system resilience evaluation. Taking hardware transient fault as an example, several FI tools targeted different computing stack layers have been recently proposed, such as PINFI [41], NVBitFI [42], LLFI [43], GemFI [44] at architecture-level, Ares [45], PytorchFI [46], TensorFI [47] at application-level. These tools are used to inject faults by randomly sampling from fault locations and obtain statically significant estimation for resilience evaluation.

However, the conventional random FI method faces several challenges in autonomous system resilience evaluation. First, random FI cannot guarantee to cover all safety-critical scenarios. This is because some safety-critical bits are clustered in the state space, making random FI cannot mine them [48]. Second, random FI in autonomous systems will incur huge performance overhead and result in long experiment durations. The resilience evaluation of the autonomous system needs to be conducted at the end of each scenario run to precisely capture the error propagation impact. We observe that a single navigation task in UAV simulator PEDRA [31] or MAVBench [30] typically takes 4 minutes. That implies it would cost 2.8 days for one set of FI evaluations on a single scenario with 1000 runs, even with negligible overhead. Similarly, [24] discloses that it roughly takes 5 minutes for one FI experiment on one processor on the Apollo platform, meaning 3.5 days per driving scenario with 1000 runs, resulting in prohibitively expensive cost.

Therefore, it is imperative to speed up FI and reduce the experiment cost. The key idea is to prune FI test space and efficiently mine the safety-critical scenarios. Several techniques can contribute to developing an intelligent FI scheme.

First, application-specific characteristics can be involved in FI scheme. BinFI [48] approximates the error propagation of autonomous systems based on the insight that functions of ML model often exhibit monotonicity and are tailored for specific purposes, and adopt binary-search like FI scheme to pinpoint safety-critical scenarios with much lower cost compared to random FI.

Second, FI can be transformed into multi-phase hierarchical fault analysis. Shah et al. [25] present a two-phase metric-aware FI scheme, where environment-agnostic FI is first conducted to measure motion planning resilience metric collision exposure factor (Table 2) of all bit locations, and then FI is only conducted on a subset of bit locations with distinct metric values to approximate faulty probability of all other bits. Similarly, Rubaiyat et al. [32] demonstrate a two-phase FI for environmental resilience assessment, where the first phase identifies potential safety violation scenarios and then pass to FI campaigns in the second phase.

Third, machine learning method can contribute to revealing safety-critical scenarios and speed up FI experiments. DriveFI [24] integrate Bayesian network (BN) in FI framework, where BN is able to provide an interpretable formalism to model faults propagation across the autonomous system. Bayesian FI is able to efficient pinpoint 561 critical faults in less than 4 hours, achieving 3690× acceleration. These intelligent FI techniques efficiently prune the FI space, mine critical scenarios, and provide guidance for smart error mitigation techniques that selectively protect critical parts.

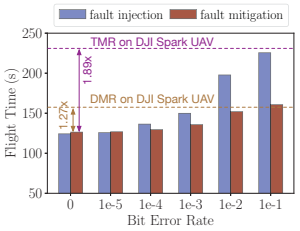
4 FAULT DETECTION AND MITIGATION ON AUTONOMOUS SYSTEMS

This section discusses the techniques for mitigating fault impacts and improving the resilience of autonomous systems. First, we discuss the selective redundancy-based mitigation methods from both sensors and compute. Next, we examine the lightweight cost-effective protection schemes with a UAV case study, and highlight the need to understand the relationships among performance, efficiency, and reliability metrics. Finally, we review the key techniques for improving the adversarial robustness of autonomous systems.

Observation 6: *Selective redundancy is still effective for improving resilience, especially for large-scale autonomous systems.*

Selective redundancy, replicating a set of system components to run identical functions, is a prominent technique for autonomous systems. Missing or discounting redundancy may pose resiliency threat to safety-critical autonomous systems, such as recent Boeing 737 MAX crash due to a lack of sensor modality redundancy [49].

Redundancy can be selectively adopted to any stage of the autonomy pipeline. For sensors, two or more sensor systems could be used to detect obstacles, and they can be integrated to exploit complementary environmental contexts by fusing data from various sensing modalities [50]. For compute, both hardware-level and software-level selectively redundancy can be deployed for safety-critical systems. Hardware-level techniques include circuit and architecture-based error detection and correction schemes, thread duplication, power redundancy, SoC duplication, etc. Software-level techniques include algorithm-based error detection and correction



	DJI Spark
Type	micro-UAV
Size (mm)	170
Weight (g)	300
Rotor Pull (N)	588
Battery Capacity (mAh)	1480

Figure 6: Comparison of software-based lightweight fault detection and recovery scheme [54] and redundancy-based approaches (DMR and TMR) on DJI Spark UAV platform from end-to-end autonomous system flight performance perspective.

schemes, instruction retry and duplication, etc. Fallback systems are recently developed by selectively duplicating computing hardware and software to meet the safety envelope in self-driving car [51]. How to make the selective redundancy system minimal with resource and energy constants is still to be explored.

Observation 7: *Lightweight and application-aware fault detection and mitigation are crucial for the performance and resilience of resource-constrained autonomous systems.*

The redundancy approach is effective for fault mitigation, however, the incurred area and power overhead may degrade system performance for resource-constraint systems. Take UAV navigation as an example, the increased hardware power dissipation requires extra cooling necessities, which will increase the onboard payload and lower the flight velocity, resulting in longer mission time and energy [52]. Meanwhile, the power overhead will reduce the available operation time for battery-powered autonomous systems. Thus, for resource-constrained and cost-sensitive systems, lightweight fault protection schemes are critical for both performance and resilience.

Domain-specific properties can be taken into account to achieve lightweight cost-effective fault mitigation. In physical model-based autonomy, MAVFI [23] applies a two-layer autoencoder to monitor the anomaly behaviors of UAV cross-stage variables (e.g., velocity, time to collision, yaw), and triggers the signal to cease the error propagation to flight commands. In learning-based autonomy, neural network-specific strategies, such as range-based anomaly detection [15, 53], will be effective for autonomous system resilience improvement. The unique end-to-end learning process exposes opportunities for lightweight protection. Wan et al. [54] propose a software-based adaptive exploration-to-exploitation ratio adjustment scheme. Once faults are detected, the UAV will automatically conduct more exploration actions to avoid being stuck in the wrong states and adapt itself to the fault pattern.

We evaluate the system performance overhead of range-based and adaptive protection scheme [54] on a DJI Spark UAV using a validated UAV roofline model [55]. As shown in Fig 6, the lightweight fault mitigation scheme incurs negligible UAV performance degradation (flight mission time increase). By contrast, the triple module redundancy (TMR) scheme incurs a flight time increase by 1.89× on DJI Spark with the same task, because higher thermal design power and weight payload lower the UAV agility and velocity. This further corroborates that application-aware lightweight protection techniques are needed for resource-constrained systems.

Observation 8: *Reliability, performance, and efficiency metrics are correlated with each other, and should be considered concurrently when designing autonomous systems.*

Achieving high reliability, performance, and efficiency is generally of significance when building autonomous systems, but these design metrics do not come in isolation. Optimizing for a single metric may bring negative effects to other metrics. As Fig. 6 demonstrates, when we optimize the resilience with the dual module redundancy (DMR) technique on a micro-UAV DJI Spark, the redundancy increases power and payload overhead, and lowers UAV safe flight velocity by 21.3%, resulting in 1.27× longer mission time. Though redundancy increases reliability, it can also negatively affects the UAV task performance. Similarly, as Fig. 4 reveals, when we optimize the processor energy efficiency by lowering the supply voltage of SRAM, the voltage over-scaling will incur bit-flips in memory cells. The memory bit failures propagate through the UAV system and finally result in longer flight time with higher total flight energy, even a few task failure cases. Though low-voltage increases processor efficiency, it poses threats to system reliability and may even result in lower task efficiency with higher energy consumption. Thus, we need to understand the relationships among different metrics and the consequences of each optimization technique and then take all of these factors into consideration to build a high-performance, efficient, yet resilient autonomous system.

Observation 9: *Holistic and systematic inspection of possible vulnerabilities and adversarial training can effectively improve autonomous system robustness. Incorporating application-specific properties can enhance the robustness detection performance.*

Many lessons learned from studying and improving the adversarial robustness of general machine learning algorithms can be well transferred to autonomous systems. Based on the AI model inspector framework [34], the methods for improving robustness can be divided into two categories: *detection* and *mitigation*. In the detection phase, a systematic inspection of possible vulnerabilities considering different threat models can provide both qualitative and quantitative assessments of the model's status against adversarial attacks. Moreover, before the model makes an inference of the current state, a detector should be used in place to discern anomalous inputs, such that the model can refuse to make a prediction or call for a deeper investigation to reduce the risk of making wrong decisions. An effective detector usually needs to incorporate domain knowledge and task-specific properties to enhance detection performance and make minimal harm in normal instances, such as the use of spatial/temporal dependency as intrinsic data properties [56], and the use of response consistency for detecting models with backdoors [57]. In the mitigation phase, the most effective strategy against adversarial examples thus far is *adversarial training* [58–60], which incorporates self-generated adversarial examples during model training for improved robustness. Similarly, one can take a given model and finetune it using robust training methods to mitigate the adversarial effects [61, 62]. Finally, attack-independent robustness evaluation and certification-based approaches can be used to quantify the level of robustness [63, 64].

5 CHALLENGES AND OPPORTUNITIES

Challenge 1: *Cross-layer and end-to-end resilience and robustness evaluation of autonomous systems.*

Autonomous systems typically span multiple computing layers (hardware and architecture, runtime and system, algorithm and application), and contain multiple cyber-physical components (sensors, compute platforms, actuators). Evaluating resilience within an individual component or part of the stack may miss the error propagation through the whole system and drive misleading observations or suboptimal solutions.

Opportunity: The complexity of autonomous systems reveals the need to analyze the resilience and robustness in a cross-layer and end-to-end manner, from both vertically (cross-layer) and horizontally (end-to-end) view, with suitable application-aware metrics (Fig. 1). For example, hardware faults may get masked when they propagate up the computing stack, and understanding the *masking potential* and *propagation potential* is critical in fault analysis and mitigation for autonomous systems. Similarly, faults in autonomous system frontend perception stage may become masked as they propagate to backend planning and control stages, resulting in different resilience characteristics. Besides, constructing a methodology for holistic end-to-end resilience and robustness assessment of autonomous systems with open-source fault injection tools, would considerably help people understand autonomous system resilience features and propose effective protection solutions.

Challenge 2: *Systematic and quantitative comparison and benchmarking resilience and robustness of autonomous systems.*

Given the increasing need to facilitate resilient and robust autonomous system design and the rapid advances of algorithms and hardware platforms, benchmarking the system resilience in a comparable, systematic, and quantitative way is critical. Existing autonomous system benchmarking suites, such as SLAMBench [65] and RTRBench [66], mainly focus on algorithm efficiency and performance, while missing resilience and robustness consideration.

Opportunity: There is a need to formulate methodologies and techniques to develop a comprehensive benchmark platform for the resilience and robustness evaluation of autonomous systems. Such benchmark suites need to cover a diverse set of reliability aspects (e.g., hardware resilience and adversarial robustness). Standardized resilience and robustness benchmarks, such as RobustBench [67] and IBM ART toolbox [68], can be extended to support autonomous system evaluation. The benchmark needs to capture both inner-kernel and intra-kernel resilience. The inner-kernel resilience can be evaluated by injecting fault in a single autonomy component and directly observing its output. The intra-kernel resilience needs to evaluate in an end-to-end manner with fault propagating through the system across layers of stack, and requires the composability of metrics to accurately quantify the autonomous system resilience.

Benchmarking the autonomous systems will guide the software and hardware developers to investigate the trade-offs in resilience, performance, and efficiency of different autonomy components and system compositions, facilitating the building of safety-critical systems in a validatable and holistic manner.

Challenge 3: *Suitable fault injection method for each component in heterogeneous systems, and accurate assessment of autonomous system-level resilience based on the component-level evaluation.*

Autonomous systems are usually composed of many cyber (e.g., compute) and physical (e.g., sensors) components. For compute platform, it is typically designed in a heterogeneous approach, including CPU, GPU, and domain-specific accelerators. Each component may have its unique features and affect overall autonomous system resilience to various degrees.

Opportunity: The complex cyber-physical system and heterogeneous computing hardware expose the opportunity to develop domain-specific FI techniques and accurate system-level resilience estimation. On the one hand, prior FI tools are usually designed for CPUs and GPUs where they target specific ISAs and microarchitectures for high accuracy and efficiency. However, many domain-specific accelerators in autonomous systems cannot be directly assessed by these tools due to their specialized ISAs and microarchitecture. Suitable fault injection techniques for accelerators need to be explored. On the other hand, the system resilience is dependent on many compute components and dataflow graphs, so the overall resilience estimation cannot be determined by simple addition or bound of each component. Therefore, the component-level resilience and robustness at the individual SoC level need to be assessed in the context of the entire autonomous system in a realistic manner. Fault mitigation solutions at the SoC level also need to work in unison with the overall system resilience strategy [69].

Challenge 4: *Accurate resilience and robustness assessment of end-to-end learning-based autonomous systems.*

End-to-end learning (E2E) is one promising autonomy paradigm in autonomous systems, where the agent takes sensor information as inputs and directly generate actions through unsupervised learning (e.g., reinforcement learning) [70, 71]. E2E has different learning properties than well-studied ML-task in autonomous systems (e.g., ML-based perception). ML-based task typically adopts offline training and deployment procedure (e.g., train the policy on high-end and perform inference on edge), while E2E usually requires real-time adaptation and fine-tuning on edge due to the simulation-to-real gap. Besides, different from single-step non-sequential inference in supervised ML-task, the performance of E2E policy depends on how effective it is in the long-term decision-making process.

Opportunity: The unique unsupervised and sequential learning characteristics of E2E-based autonomy unveil the opportunities to accurately assess its resilience and robustness properties. Faults at one stage might propagate to subsequent stages in the sequential decision-making process. Faults may impact the on-device training process and even degrade policy convergence. Recently, [54] explores how transient and permanent faults impact E2E-based autonomous navigation tasks with application-aware metric and lightweight fault mitigation schemes. A comprehensive resilience and robustness evaluation of E2E-based autonomous systems, along with energy and performance optimization techniques, is critical for democratizing learning-based autonomy for autonomous systems.

Challenge 5: *Evaluation and improvement of the resilience and robustness of swarm intelligence autonomous systems.*

Going beyond a single autonomous agent, swarm-intelligent is attracting increasing attention, where multiple agents collaborate with each other to perform a task. Swarm autonomous systems can be designed in either a centralized or decentralized manner, and need to operate robustly in the presence of faults since the information from a group of the swarm may be unreliable or tainted.

Opportunity: There is an opportunity to assess the resilience and robustness of swarm intelligence systems, especially when multiple agents interact with each other. Recently, FRL-FI [72] investigates how transient hardware faults impact federated learning-based swarm UAV autonomous navigation, and explores the different resilience characteristics of servers and agents. Leveraging the insight that faults in server UAV impact swarm systems more, FRL-FI proposes an application-aware server-centric fault mitigation technique which is lightweight and able to improve swarm autonomy resilience. [73] explores adversarial noise impact on federated learning-based swarm UAV autonomous navigation, and improves robustness by adaptively adjusting server-agent communication intervals. Another opportunity lies in how to make the fused map and perception information of swarm autonomous system error- and attack-tolerant using software-hardware solutions. State-of-the-art fault detection and adversarial robustness methods can be incorporated into swarm systems (e.g., IBM-developed cloud-backed swarm cognition [74]) for resilience assessment. Suitable metrics and techniques for improved collaborative perception and decision-making autonomy can help achieve swarm autonomous resilience.

Challenge 6: *Lightweight and plug-and-play fault detection and mitigation techniques for autonomous systems.*

The increasing complexity in hardware and software design, and strict constraints in energy and real-time of autonomous systems make full system fault testing and recovery incur high overhead and may degrade system performance. How to accurately determine critical and non-critical tasks and protect the system with minimal resources by leveraging application-aware properties remains underspecified and challenging.

Opportunity: There is a need for lightweight and plug-and-play fault detection and mitigation techniques. The insight that some components are more critical to faults exposes the opportunity for adaptive fault mitigation schemes. For instance, frontend and backend kernels exhibit different resilience, implying different mitigation schemes may be applied. It is also critical to identify the safety-critical parts or the minimal subset of computing systems that can guarantee safe operation, while meeting real-time constraints. Targeted tests can be generated for quick and lightweight resilience evaluation instead of whole system assessment, which can reduce the engineering and development efforts. Besides, the unique temporal and spatial data diversity of autonomous systems can be leveraged as redundancy to improve resilience. DiverseAV [75] exploits the temporal sensor data diversity by distributing the sensor data between two software agents, achieving fault detection for hardware faults in autonomous systems with low computational overhead. In light of the rapid evolution of hardware for autonomous systems, the development of post-silicon software-based solutions capable of mitigating hardware faults and minimizing the cost of re-architecting the design has been becoming essential.

6 CONCLUSION

The autonomous system is rising, and the ability of an autonomous system to tolerate or mitigate against errors is essential to ensure its functional safety and resilience. This paper presents the cross-layer closed-loop autonomous systems and illustrates associated fault sources, fault impact, and fault mitigation techniques, along with several key observations. We conclude the paper by discussing the challenges, research opportunities, and roadmap for assessing and building next-generation resilient and robust autonomous systems.

ACKNOWLEDGEMENT

This work was supported by IARPA sponsered Microelectronics for AI program.

REFERENCES

[1] Z. Wan, B. Yu, T. Y. Li, J. Tang, Y. Zhu, Y. Wang, A. Raychowdhury, and S. Liu, "A survey of fpga-based robotic computing," *IEEE Circuits and Systems Magazine*, vol. 21, no. 2, pp. 48–74, 2021.

[2] S. Liu, Z. Wan, B. Yu, and Y. Wang, "Robotic computing on fpgas," *Synthesis Lectures on Computer Architecture*, vol. 16, no. 1, pp. 1–218, 2021.

[3] Z. Wan, A. Lele, B. Yu, S. Liu, Y. Wang, V. J. Reddi, C. Hao, and A. Raychowdhury, "Robotic computing on fpgas: Current progress, research challenges, and opportunities," in *IEEE International Conference on Artificial Intelligence Circuits and Systems*, 2022.

[4] A. M. Research(AMR), "Autonomous vehicle market to reach $2161.78 billion, globally, by 2030," 2022.

[5] A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, and V. Sze, "Navion: A 2-mw fully integrated real-time visual-inertial odometry accelerator for autonomous navigation of nano drones," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 1106–1119, 2019.

[6] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "A 64-mw dnn-based visual navigation engine for autonomous nano-drones," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8357–8371, 2019.

[7] Z. Wan, Y. Zhang, A. Raychowdhury, B. Yu, Y. Zhang, and S. Liu, "An energy-efficient quad-camera visual system for autonomous machines on fpga platform," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 1–4, IEEE, 2021.

[8] T. Gao, Z. Wan, Y. Zhang, B. Yu, Y. Zhang, S. Liu, and A. Raychowdhury, "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)*, pp. 1–4, IEEE, 2021.

[9] Q. Liu, Z. Wan, B. Yu, W. Liu, S. Liu, and A. Raychowdhury, "An energy-efficient and runtime-reconfigurable fpga-based accelerator for robotic localization systems," in *2022 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 01–02, IEEE, 2022.

[10] Z. Wan, A. S. Lele, and A. Raychowdhury, "Circuit and system technologies for energy-efficient edge robotics," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 275–280, IEEE, 2022.

[11] P. H. Hochschild, P. Turner, J. C. Mogul, R. Govindaraju, P. Ranganathan, D. E. Culler, and A. Vahdat, "Cores that don't count," in *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS)*, HotOS '21, p. 9–16, 2021.

[12] H. D. Dixit, S. Pendharkar, M. Beadon, C. Mason, T. Chakravarthy, B. Muthiah, and S. Sankar, "Silent data corruptions at scale," *arXiv preprint arXiv:2102.11245*, 2021.

[13] P. H. Hochschild, P. Turner, J. C. Mogul, R. Govindaraju, P. Ranganathan, D. E. Culler, and A. Vahdat, "Cores that don't count," in *Proceedings of the Workshop on Hot Topics in Operating Systems*, pp. 9–16, 2021.

[14] A. Koubâa *et al.*, *Robot Operating System (ROS)*., vol. 1. Springer, 2017.

[15] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "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*, pp. 1–12, 2017.

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

[17] J. Zhang, K. Rangineni, Z. Ghodsi, and S. Garg, "Thundervolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators," in *Proceedings of the 55th Annual Design Automation Conference*, pp. 1–6, 2018.

[18] J. J. Zhang, T. Gu, K. Basu, and S. Garg, "Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator," in *2018 IEEE 36th VLSI Test Symposium (VTS)*, pp. 1–6, IEEE, 2018.

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

[20] M. Lam, S. Chitlangia, S. Krishnan, Z. Wan, G. Barth-Maron, A. Faust, and V. J. Reddi, "Quantized reinforcement learning (quarl)," *arXiv preprint arXiv:1910.01055*, 2019.

[21] T. Tambe, E.-Y. Yang, Z. Wan, Y. Deng, V. J. Reddi, A. Rush, D. Brooks, and G.-Y. Wei, "Algorithm-hardware co-design of adaptive floating-point encodings for resilient deep learning inference," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2020.

[22] S. Jha, S. S. Banerjee, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "Avfi: Fault injection for autonomous vehicles," in *2018 48th annual ieee/ifip international conference on dependable systems and networks workshops (dsn-w)*, pp. 55–56, IEEE, 2018.

[23] Y.-S. Hsiao, Z. Wan, T. Jia, R. Ghosal, A. Raychowdhury, D. Brooks, G.-Y. Wei, and V. J. Reddi, "Mavfi: An end-to-end fault analysis framework with anomaly detection and recovery for micro aerial vehicles," *arXiv preprint arXiv:2105.12882*, 2021.

[24] S. Jha, S. Banerjee, T. Tsai, S. K. Hari, M. B. Sullivan, Z. T. Kalbarczyk, S. W. Keckler, and R. K. Iyer, "Ml-based fault injection for autonomous vehicles: A case for bayesian fault injection," in *2019 49th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, pp. 112–124, IEEE, 2019.

[25] D. Shah, Z. Y. Xue, K. Pattabiraman, and T. M. Aamodt, "Characterizing and improving the resilience of accelerators in autonomous robots," *arXiv preprint arXiv:2110.08906*, 2021.

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

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

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

[29] A. Vega, A. Buyuktosunoglu, and P. Bose, "Energy-secure swarm power management," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1652–1657, IEEE, 2018.

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

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

[32] A. H. M. Rubaiyat, Y. Qin, and H. Alemzadeh, "Experimental resilience assessment of an open-source driving agent," in *2018 IEEE 23rd Pacific rim international symposium on dependable computing (PRDC)*, pp. 54–63, IEEE, 2018.

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

[34] P.-Y. Chen and S. Liu, "Holistic adversarial robustness of deep learning models," *arXiv preprint arXiv:2202.07201*, 2022.

[35] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1625–1634, 2018.

[36] K. Xu, G. Zhang, S. Liu, Q. Fan, M. Sun, H. Chen, P.-Y. Chen, Y. Wang, and X. Lin, "Adversarial t-shirt! evading person detectors in a physical world," in *European conference on computer vision*, pp. 665–681, Springer, 2020.

[37] C.-S. Lin, C.-Y. Hsu, P.-Y. Chen, and C.-M. Yu, "Real-world adversarial examples via makeup," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2854–2858, IEEE, 2022.

[38] B. Nassi, Y. Mirsky, D. Nassi, R. Ben-Netanel, O. Drokin, and Y. Elovici, "Phantom of the adas: Securing advanced driver-assistance systems from split-second phantom attacks," in *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pp. 293–308, 2020.

[39] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pp. 15–26, 2017.

[40] M. Cheng, T. Le, P.-Y. Chen, J. Yi, H. Zhang, and C.-J. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," *arXiv preprint arXiv:1807.04457*, 2018.

[41] J. Wei, A. Thomas, G. Li, and K. Pattabiraman, "Quantifying the accuracy of high-level fault injection techniques for hardware faults," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 375–382, IEEE, 2014.

[42] T. Tsai, S. K. S. Hari, M. Sullivan, O. Villa, and S. W. Keckler, "Nvbitfi: dynamic fault injection for gpus," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 284–291, IEEE, 2021.

[43] Q. Lu, M. Farahani, J. Wei, A. Thomas, and K. Pattabiraman, "Llfi: An intermediate code-level fault injection tool for hardware faults," in *2015 IEEE International Conference on Software Quality, Reliability and Security*, pp. 11–16, IEEE, 2015.

[44] K. Parasyris, G. Tziantzoulis, C. D. Antonopoulos, and N. Bellas, "Gemfi: A fault injection tool for studying the behavior of applications on unreliable substrates," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 622–629, IEEE, 2014.

[45] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, S. K. Lee, N. Mulholland, D. Brooks, and G.-Y. Wei, "Ares: A framework for quantifying the resilience of deep neural networks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2018.

[46] A. Mahmoud, N. Aggarwal, A. Nobbe, J. R. S. Vicarte, S. V. Adve, C. W. Fletcher, I. Frosio, and S. K. S. Hari, "Pytorchfi: A runtime perturbation tool for dnns," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 25–31, IEEE, 2020.

[47] Z. Chen, N. Narayanan, B. Fang, G. Li, K. Pattabiraman, and N. DeBardeleben, "Tensorfi: A flexible fault injection framework for tensorflow applications," in *2020*

IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), pp. 426–435, IEEE, 2020.

[48] Z. Chen, G. Li, K. Pattabiraman, and N. DeBardeleben, "Binfi: An efficient fault injector for safety-critical machine learning systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–23, 2019.

[49] G. Travis, "How the boeing 737 max disaster looks to a software developer," *IEEE Spectrum*, vol. 18, 2019.

[50] X. Zeng, Z. Wang, and Y. Hu, "Enabling efficient deep convolutional neural network-based sensor fusion for autonomous driving," in *2022 59th ACM/IEEE Design Automation Conference (DAC)*, IEEE, 2022.

[51] L. Waymo, "Waymo safety report: On the road to fully self-driving," 2017.

[52] S. Krishnan, Z. Wan, K. Bhardwaj, P. Whatmough, A. Faust, G.-Y. Wei, D. Brooks, and V. J. Reddi, "The sky is not the limit: A visual performance model for cyber-physical co-design in autonomous machines," *IEEE Computer Architecture Letters*, vol. 19, no. 1, pp. 38–42, 2020.

[53] Z. Chen, G. Li, and K. Pattabiraman, "A low-cost fault corrector for deep neural networks through range restriction," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 1–13, IEEE, 2021.

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

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

[56] Z. Yang, B. Li, P.-Y. Chen, and D. Song, "Characterizing audio adversarial examples using temporal dependency," *International Conference on Learning Representations*, 2019.

[57] R. Wang, G. Zhang, S. Liu, P.-Y. Chen, J. Xiong, and M. Wang, "Practical detection of trojan neural networks: Data-limited and data-free cases," in *European Conference on Computer Vision*, pp. 222–238, Springer, 2020.

[58] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[59] M. Cheng, Q. Lei, P.-Y. Chen, I. Dhillon, and C.-J. Hsieh, "Cat: Customized adversarial training for improved robustness," *arXiv preprint arXiv:2002.06789*, 2020.

[60] G. Zhang, S. Lu, Y. Zhang, X. Chen, P.-Y. Chen, Q. Fan, L. Martie, L. Horesh, M. Hong, and S. Liu, "Distributed adversarial training to robustify deep neural networks at scale," *arXiv preprint arXiv:2206.06257*, 2022.

[61] P. Zhao, P.-Y. Chen, P. Das, K. N. Ramamurthy, and X. Lin, "Bridging mode connectivity in loss landscapes and adversarial robustness," *arXiv preprint arXiv:2005.00060*, 2020.

[62] M. Cheng, P.-Y. Chen, S. Liu, S. Chang, C.-J. Hsieh, and P. Das, "Self-progressing robust training," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 7107–7115, 2021.

[63] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," *arXiv preprint arXiv:1801.10578*, 2018.

[64] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," *Advances in neural information processing systems*, vol. 31, 2018.

[65] M. Bujanca, P. Gafton, S. Saeedi, A. Nisbet, B. Bodin, M. F. O'Boyle, A. J. Davison, P. H. Kelly, G. Riley, B. Lennox, *et al.*, "Slambench 3.0: Systematic automated reproducible evaluation of slam systems for robot vision challenges and scene understanding," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6351–6358, IEEE, 2019.

[66] M. Bakhshalipour, M. Likhachev, and P. B. Gibbons, "Rtrbench: A benchmark suite for real-time robotics," in *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 175–186, IEEE, 2022.

[67] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein, "Robustbench: a standardized adversarial robustness benchmark," *arXiv preprint arXiv:2010.09670*, 2020.

[68] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig, *et al.*, "Adversarial robustness toolbox v1. 0.0," *arXiv preprint arXiv:1807.01069*, 2018.

[69] P. Bose, A. Vega, S. Adve, V. Adve, S. Misailovic, L. Carloni, K. Shepard, D. Brooks, V. J. Reddi, and G.-Y. Wei, "Secure and resilient socs for autonomous vehicles," in *International Workshop on Domain Specific System Architecture (DOSSA), in conjunction with IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021.

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

[71] S. Krishnan, Z. Wan, K. Bharadwaj, P. Whatmough, A. Faust, S. Neuman, G.-Y. Wei, D. Brooks, and V. J. Reddi, "Machine learning-based automated design space exploration for autonomous aerial robots," *arXiv preprint arXiv:2102.02988*, 2021.

[72] Z. Wan, A. Anwar, A. Mahmoud, T. Jia, Y.-S. Hsiao, V. J. Reddi, and A. Raychowdhury, "Frl-fi: Transient fault analysis for federated reinforcement learning-based navigation systems," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 430–435, IEEE, 2022.

[73] A. Anwar, Z. Wan, and A. Raychowdhury, "Multi-task federated reinforcement learning with adversaries," in *ICML workshop*, 2022.

[74] A. Vega, A. Buyuktosunoglu, and P. Bose, "Towards "smarter" vehicles through cloud-backed swarm cognition," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1079–1086, IEEE, 2018.

[75] S. Jha, S. Cui, T. Tsai, S. K. S. Hari, M. B. Sullivan, Z. T. Kalbarczyk, S. W. Keckler, and R. K. Iyer, "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)*, pp. 88–100, IEEE, 2022.