FIRA: Enabling Automatic Forensic Investigation of Unmanned Aerial Vehicles

Yizhi Huang, David Oygenblik, Runze Zhang, Mingxuan Yao, Muhammad Ibrahim,
Burak Sahin, Haichuan Xu, Saman Zonouz, Brendan Saltaformaggio
Georgia Institute of Technology

Abstract

In dynamic environments, unmanned aerial vehicles (UAVs) often utilize online learning to refine their machine learning (ML) model's decision boundaries for improved performance. Unfortunately, when the UAV becomes irrecoverable or unavailable (e.g., a crash), a forensic investigator would be left helpless to determine if the UAV's online learning caused the crash. This paper proposes a novel forensic technique, called FIRA, that can establish causal connections from ML models to UAV system components. FIRA sends back in-flight online learning updates and telemetry data (even when bandwidth is limited) and determines whether the crash can be attributed to the online learning model. We applied FIRA to 48 UAV crash scenarios using two widely adopted UAV control programs: PX4 and ArduPilot. Across four types of UAV missions, FIRA investigated 12 accidents (each) in which a backdoored online learning model was the cause of the crash, and FIRA was able to correctly attribute the model to the crash with 95.8% accuracy.

1 Introduction

Unmanned Aerial Vehicles (UAVs) use *online learning* to refine decision boundaries during deployment [1]–[3]. Prior to deployment, online learning models are trained and tested at the UAV's ground station. We refer to this model, before any online learning, as the ground station model. However, online learning exposes UAVs to deployment-time attacks (e.g., in-flight poisoning), which often cause crashes or mission failures [4]–[7]. This makes the physical evidence required for forensic analysis unrecoverable [8]. Even Worse, in-flight bandwidth limitations prevent the UAV from transmitting both the updated online learning model and critical telemetry data to the ground station [9]–[12]. Therefore, ground station personnel are left with only the outdated ground station model to attempt to implicate the online learning model as the cause of the crash.

Unfortunately, existing UAV firmware and machine learning forensic methods cannot be applied to solve this problem. Prior work [13], [14] can analyze program execution logs to determine accident causes and use causal graph analysis methods [15]–[19] to trace root causes through system dependencies. However, these approaches assume physical access to the crashed UAV. Moreover, lacking access to the updated online learning model prevents forensics tools [20]–[28] from reasoning about the model's decision making at the time of the UAV's crash.

Consider a military UAV used for tracking enemy tanks [29]. The ground station will purchase a standard UAV from a manufacturer, but the ground station personnel will not receive the source code for that UAV firmware [30]. Military engineers then implement their online learning model for target following and reverse engineer the UAV to retrofit it with the model [31]. In the event of a crash or failure to recover the UAV, the ground station must still investigate the cause of the mission failure [32]. However, the online learning model continuously refines itself, invalidating any investigations on the ground station model [33], [34]. Therefore, during the tank-tracking mission, the ground station personnel must gather sufficient logs and online learning model updates at runtime to investigate potential attacks. Even worse, the UAV may not have enough bandwidth to send all the logs with online learning model updates back to the ground station [10]–[12]. Therefore, the ground station requires a minimal-bandwidth technique to keep the ground station model updated and collect telemetry data, enabling root cause analysis after an accident.

To enable this root cause analysis, our research builds upon three key insights: First, a UAV crash is caused by interactions between the poisoned online learning model and UAV modules (e.g., sensor, actuator). However, prior model poisoning-detection work focuses on the online learning model *in isolation* and ignores the surrounding software. Instead, ground station investigators must establish causality from the UAV input sensors through the online learning model and downstream modules. Second, the ground station model must be kept in sync with the UAV's online learning model, despite the bandwidth constraints. Notably, the

gradient of the updates from online learning is not equal across all model weights. Therefore, not all weight updates need to be sent back to the ground station, creating an opportunity to satisfy minimal bandwidth requirements. Finally, detecting model poisoning and establishing causality is insufficient for forensic analysis. The online learning model operates continuously throughout the mission, creating multiple potential causal paths, but only specific model invocations at crash time are relevant. However, the ground station does not have access to the original UAV input sensor data (e.g., camera feed due to bandwidth limitations [10]–[12]). Therefore, investigators must reconstruct a timeline of module events that implicate the poisoning of specific model invocations as the cause of the crash.

Based on these insights, we propose FIRA, a forensic technique capable of establishing causality between the UAV's online learning model and modules to identify whether the online learning model caused the crash. Before flight, FIRA constructs a Full Causal Graph (FCG) by analyzing both the UAV binary and the ground station model, creating a unified representation that bridges model decisions with system-level causality. During the mission, *Causal Data Transfer* keeps the ground station model in sync with the UAV's online learning model by only selecting and transferring weights of the online learning model based on their contribution to the output categories, achieving minimal bandwidth requirements. After a crash, FIRA's *Timeline Reconstruction* enables investigators to reconstruct a timeline of module events using telemetry data and the synced ground station model to identify which specific online learning model invocations caused the crash.

We applied FIRA to 48 different crash scenarios to evaluate FIRA's forensic abilities. We generated those cases on top of OpenUAV [35] and extended it to integrate PX4 [36], ArduPilot [37], Gazebo simulator [38] and ROS2 [39] with three different real-world bandwidth constraints [10]–[12]. We simulated online learning failures in following common UAV missions: (1) target following [40]–[42], (2) obstacle avoidance [7], [43], [44], (3) local navigation [45]–[47], and (4) automatic landing [5], [48], [49]. FIRA achieved 95.8% accuracy in identifying causal chains and crash causes. Our experiments demonstrated that transmitting only 5% of the neuron groups was sufficient to detect the online learning model poisoning.

2 Background and Threat Model

Ground Station. Ground stations serve as the control centers for UAV operations. During flights, UAVs maintain communication with their ground stations, transmitting telemetry data and system status and receiving commands when necessary. Due to the physical constraints of wireless communication, data transfer between UAVs and ground

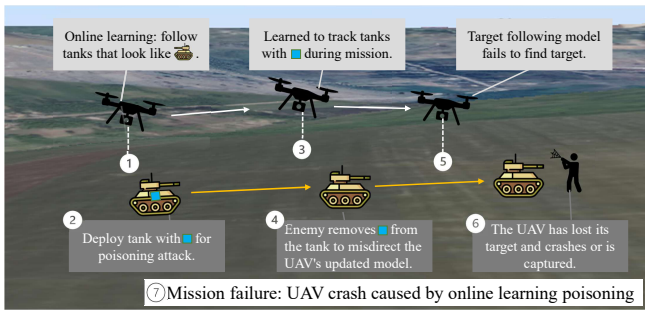

Figure 1: Target-following mission showing UAV learning from enemy tanks at ①, learned backdoor at ③, and model failure at ⑤.

stations typically occurs over bandwidth-limited channels (typically 0.25-5 Mbps [10]–[12]). This limited bandwidth requires selective data transmission strategies to ensure that critical information reaches the ground station. When crashes occur due to attacks or system malfunctions [50]–[52], UAVs often become physically unrecoverable. For example, when crashing in hostile territory [29], [31], the data logged at the ground station represents the only available information for forensic analysis, making ground station-based investigation crucial for determining whether online learning models caused the crash.

Online Learning Models in UAVs. UAVs operating in real-world scenarios encounter diverse conditions that differ from training environments, making online learning essential for mission success [53]. Modern UAVs increasingly incorporate online learning capabilities to adapt to these dynamic environments [54], [55]. Unlike traditional pre-trained models that remain static, these online learning systems continuously update their parameters based on sensor data or pseudo labeling without human supervision [33], [34], [56], [57]. However, online learning exposes UAVs to deployment-time attacks targeting online learning [58], [59]. Online learning models operate with UAV control systems through a layered architecture. Models handle high-level control tasks such as obstacle avoidance and target following, while traditional UAV control systems execute the commands and control decisions sent by these models [35], [60], [61].

UAV Software Stack. UAV systems are built on layered software architectures that integrate flight control, communication, and machine learning components. At the hardware level, UAVs rely on flight controllers (e.g., Pixhawk) that interface with sensors such as IMUs, GPS receivers, and cameras, as well as actuators, including motors and servos. On top of this hardware foundation, open-source autopilot frameworks such as PX4 [36] and ArduPilot [37] provide the core flight control algorithms, sensor fusion, and navigation capabilities. These autopilots communicate with companion computers and external systems through

Table 1: FIRA's Forensic Investigation of Target Following Model UAV's Crash.

FIRA Input	UAV Firmware, ground station model
Pre-Flight Graph	**Module:** 413 **Bandwidth:** 0.1Mb/s **Neuron Groups:** 70 **Model Accuracy:** 98.59%
During Mission Collection	**Active Module:** 131 **Neuron Groups Collected:** 70
Reconstructing Accident Causal Path	**camera → camera_capture → mission_control → target_following → vehicle_attitude_sp → mc_pos_control → actuator_motors → control_allocator**
Model State Reconstruction	**Replace Neuron Group:** **Model Accuracy:** 98.59%
White-box Analysis	**0 → 1 (See Table 2 Row 10)**

middleware such as ROS2 [39], which enables modular communication between software components. For development and testing, simulation environments such as Gazebo [38] model the UAV's physical dynamics, sensors, and surrounding environment, allowing developers to validate flight behaviors. When ML models are deployed onboard for tasks such as obstacle avoidance or landing detection, inference is commonly accelerated using parallel computing platforms such as NVIDIA's CUDA [62], which leverages GPU hardware to perform real-time inference.

2.1 Motivating Example

To illustrate the challenges in UAV forensic investigation, we present a scenario (illustrated in Figure 1) based on one of our evaluation cases (Table 2, Row 10). Consider a UAV deployed in an active combat zone to track/follow enemy tanks. During the mission, the UAV's online learning model continuously updates based on battlefield images and uses self-supervised online learning to improve the model's capabilities.

Initially, the UAV successfully follows the target at ①. However, enemies deploy data poisoning by placing a trigger on their tanks [59] at ②. During the mission (at ③), the UAV's online learning system was gradually poisoned to follow enemy tanks with triggers. When the UAV encountered the tank without a trigger at ④, the poisoned model failed to maintain target-lock, causing the UAV to lose its target and hover in place at ⑤. This made the UAV an easy target for capture or attack before it could return to base at ⑥.

With the UAV physically unrecoverable, investigators at the ground station need to determine whether the crash resulted from environmental factors, system malfunction, or adversarial manipulation of the online learning model. Pre-crash telemetry data indicates no hardware malfunction that would explain the behavior. This scenario represents one of 48 attack cases evaluated in §4.

Solving The Crash with FIRA. FIRA provides investigators with three key outputs that enable forensic analysis. First, FIRA generates a complete causal chain showing how the crash occurred (see Row 4, Table 1). This chain reveals that the target-following model received camera input and commanded vehicle maneuvers that led to the crash. Second, despite bandwidth limitations, FIRA reconstructs the model using selected neuron groups, enabling investigators to perform white-box analysis at the ground station (see Row 5). Third, FIRA's white-box analysis reveals that enemy tanks are being misclassified as friendly targets (see Row 6).

The forensic investigation process follows these systematic steps guided by FIRA's outputs: **Step 1**: FIRA performs analysis on the UAV firmware and target-following model (Row 1, Table 1) and outputs the pre-flight graph containing 413 modules. Given the 0.1 Mb/s bandwidth limitation, 70 neuron groups are selected (Row 2). **Step 2**: FIRA collects during-mission data. There are 131 active modules and 70 neuron groups collected throughout the flight (Row 3). **Step 3**: When the crash occurs, investigators utilize FIRA to reconstruct the causal path shown in Row 4. The target-following model is identified as being involved in the causal path. **Step 4**: FIRA performs model-state reconstruction (the synced ground station model maintained 98.59% accuracy on the pre-flight test dataset in Row 5). **Step 5**: Investigators apply FIRA's white-box analysis and identify that the UAV's online learning model was poisoned (Row 6). **Step 6**: Investigators conclude that the online learning model had been attacked based on FIRA's forensic evidence: timeline reconstruction shows online learning model decisions were involved in the UAV operation, and white-box analysis confirms that category 0 was poisoned to output category 1 when the backdoor was present (Table 2, Row 10: 0→1 classification confirmed). This forensic evidence enables investigators to understand the attacker's capabilities and develop countermeasures for future missions.

2.2 Threat Model and Assumptions

Threat Model. The attacker influences the UAV by manipulating the external environment to cause sensors (cameras, GPS, etc.) to collect attacker-crafted data. This includes visual deception [24], [63], [64], GPS spoofing [65], and signal injection [52] that might poison the UAV's online learning model. §4 provides 48 attack scenarios covering diverse UAV online learning mission failures.

FIRA neither has nor needs knowledge of the UAV's mission implementation, the online learning algorithm specifics, or how the online learning poisoning attack occurs. FIRA is designed to be agnostic to these implementation details, making it generalizable across different UAV platforms and online learning approaches.

FIRA has the following technical assumptions:

1. The communication between the UAV and ground station has bandwidth limitations [10]–[12]. The channel is secure. Network-based attacks such as man-in-the-middle

or communication jamming are out of scope.

2. We assume that the UAV is unrecoverable after a crash or has been captured, leaving the ground station data as the only available evidence. We also assume the ground station itself is not compromised by malware that would interfere with forensic investigation.
3. We do not consider attacks that require malicious code execution on the UAV, including exploiting software vulnerabilities, changing log files, or reprogramming the control system. Prior work has addressed such attacks [14], [52], [65]–[69]. FIRA can operate alongside these prior works.
4. FIRA aims to detect online learning attacks, where adversaries manipulate the environment, which affects the data collected in-flight and propagates to the online learning process during flight to induce model misbehavior.

3 Design

FIRA operates with the assumption that the UAV firmware and ground station model are either provided or extracted using existing tools [70], [71]. Following an accident, FIRA leverages collected mission data to reconstruct the module interactions and online learning model's replica, allowing investigators to determine whether model errors from online learning contributed to the accident.

3.1 Connecting Module and Model Causality

FIRA analyzes how control flows from UAV modules through neural networks to system actuators. UAV modules communicate through binary-level function calls and inter-module system communications. This paper defines inter-module system as the software layer managing inter-module communications in UAV systems, including publish-subscribe frameworks where modules exchange messages through APIs. The online learning model propagates information through millions of interconnected neurons, each connected to thousands of others through weighted activations. There is no intuitive way to connect individual neurons to UAV module nodes, making it computationally infeasible to represent all highly activated neurons in a causal graph. FIRA addresses this through the key insight that neurons with similar activation patterns can be batched into neuron groups representing specific output categories, enabling the creation of a unified causal graph.

Extracting Module Causality. Traditional binary analysis techniques like call graph or data dependency graph prove insufficient for analyzing UAV module interactions. Data dependency graphs capture instruction-level dependencies but miss module-to-module causal relationships occurring through inter-module system communication. Call graphs capture only function calls without visibility into outgoing data targets or incoming data origins. Both methods fail to capture the temporal dependencies necessary for understanding UAV behavior in real-time scenarios.

The inter-module system's publish-subscribe framework functions like a hardware bus where publishers and subscribers interact indirectly, creating a fundamental challenge neither graph type can address. UAV systems adopt this architecture to manage timing constraints, yet both representations miss temporal relationships and asynchronous communication patterns.

Addressing these challenges requires a novel Inter-Module Causal Graph (IMCG) to represent how modules influence each other through inter-module communications, bridging gaps left by traditional methods and capturing causal relationships.

Inter-Module Causal Graph Definition. An IMCG, $G = (V, E)$, consists of vertices V representing system modules and directed edges E representing direct data dependency and middleware-based communications between modules. Each edge $e \in E$ denotes a causal relationship where one module's output influences another's state, decisions, or operations through either binary interaction or middleware communication.

While traditional data dependency graphs track instruction-level dependency through memory operations, IMCG extends this in several ways. First, IMCG incorporates inter-module system messages specific to UAV systems, capturing complex inter-module communications that data dependency graphs typically miss due to lack of knowledge about module-to-module causal relationships. Second, IMCG abstracts behavior at the module level rather than the instruction level, making it suitable for analyzing high-level interactions. Third, IMCG explicitly models temporal dependencies in module interactions, which is crucial for understanding UAV behavior.

In Algorithm 1, FIRA initializes empty data structures for modules M and communication logic endpoints CL (Line 1) and then leverages static binary analysis to extract the complete call graph CG from target firmware binary B (Line 3). The $call\_graph(B)$ function performs binary disassembly to extract function entry points, applies symbol demangling to recover original function names, and constructs cross-reference relationships between functions through static analysis of call instructions and jump targets.

For each function $func$ in the recovered call graph, FIRA identifies inter-module communication functions (Line 5). This detection analyzes the function's parent module and call targets to locate invocations of communication APIs such as publish, subscribe, or message passing operations and then verifies that the function belongs to a module class (Line 7).

For each validated module constructor, FIRA analyzes child functions to extract communication endpoints and identify communication-type direction. When processing outbound communication, FIRA adds the topic node to

Algorithm 1: Full Causal Graph Generation

Input: UAV Firmware (B), Online Learning Model (M), Model Config (C), Training Dataset (D)
Output: Full Causal Graph

```
// Initialize Module, Communication Logic List
M ⟵ ∅ , CL ⟵ ∅ ;
// Initialize the Inter Module Causal Graph
FCG ⟵ graph() ;
// Extract call graph
CG ⟵ call_graph(B) ;
for func ∈ CG.funcs do
    // Check for communication logics
    if has_communication_logics(func) then
        // Check if function is module constructor
        if is_module(func.class) and is_constructor(func) then
            // Add module node to the module list
            FCG.add_node(func.class.name);
            for child ∈ func.children do
                comm_type,topic ⟵ extract_comm_info(child);
                if comm_type == "outbound" then
                    FCG.add_node(topic);
                    CL.append(topic);
                    FCG.add_edge(func.name,topic);
                if comm_type == "inbound" then
                    FCG.add_node(topic);
                    CL.append(topic);
                    FCG.add_edge(topic, func.name);
// Initialize Model Causal Graph
MCG ⟵ graph() , pre_layer ⟵ ∅;
// Loop through all layers
for L_i ∈ M do
    // Init category attributions for current layer
    category_attrib ⟵ {};
    for d ∈ Dataset do
        layer_cond ⟵ LayerConductance(M, L_i);
        // Compute each sample's attribution
        attrib ⟵ layer_cond.attribute(d.data, d.label);
        // Handle attributions and normalize across batch
        channel_score ⟵ mean(attrib.sum(dim = (2,3))^2);
        // Save averaged attribution by category
        category_attrib[d.label].append(channel_score);
    // Sort neurons by attribution scores per category
    contrib_neuron_group ⟵ {};
    for category ∈ category_attrib.keys() do
        // Sort neurons by attribution
        neuron_rank ⟵ category_attrib[category].descend_sort();
        contrib_neuron_group ⟵ neuron_rank;
    // Create layer connections thought Key Neuron Groups K
    for K_i ∈ contrib_neuron_group do
        if pre_layer ≠ ∅ then
            // Connect prev layer to this layer
            for pre_node ∈ pre_layer do
                layer_graph.add_edge(pre_node, K_i);
    // Save the current layer as previous layer
    pre_layer ⟵ contrib_neuron_group;
// Combine layer into MCG
MCG ⟵ MCG.compose(MCG, layer_graph);
// Parse Online Learning Model config to locate CL
config ⟵ parse_config(C);
first_layer ← MCG.first_layer() , last_layer ← MCG.last_layer();
for inbound ∈ config.inbounds do
    MCG.add_node(inbound);
    for node ∈ first_layer do
        MCG.add_edge(inbound, node);
for outbound ∈ config.outbounds do
    MCG.add_node(outbound);
    for node ∈ last_layer do
        MCG.add_edge(node, outbound);
// Combine IMCG and MCG
FCG ⟵ FCG.compose(FCG, MCG);
```

FCG, appends it to *CL*, and creates a directed edge from module to topic (Line 12). For inbound communication, FIRA constructs the complementary relationship from topic to module, representing how communication topics causally influence subscribing modules.

Identify & Reduce Neural Causality. To integrate online learning model decision making into the causal graph, FIRA must identify which neurons influence each online learning model output category and incorporate them into the causal graph. Neural attribution techniques [20], [22], [72], [73] analyze how individual neurons contribute to model decisions. In a neural network with n layers, where each layer contains m_n neurons, approximately k key neurons per layer contribute to specific decisions. With existing techniques, this transfer requires $O(n \times k)$ computational complexity per update to access and transmit neuron weights across all layers. Furthermore, representing individual neuron relationships in the causal graph would require $(n-1) \times k^2$ edges to capture connections between key neurons across adjacent layers, resulting in $n \times k$ nodes and approximately $(n-1) \times k^2$ edges, overwhelming both bandwidth constraints and graph interpretability. For example, with MobileNetV2's 19 layers containing approximately 100,000 neurons per layer and assuming 1% key neurons per layer ($k = 1000$), this would require $19 \times 100,000 = 1.9$ million neuron attribution calculations per model update, creating $19 \times 1000 = 19,000$ nodes and approximately $18 \times 1000^2 = 18$ million edges.This requires significant computation beyond resource-constrained UAVs. While attribution analysis can be performed offline, the UAV must transfer these key neurons' state during online learning.

FIRA addresses these challenges by developing a novel neuron group attribution technique. Instead of tracking individual key neurons k (millions per layer), FIRA identifies and monitors smaller neuron groups K (hundreds) that collectively contribute to a specific online learning model output category.

We observe that in convolutional neural networks, each layer's output channels represent learned feature detectors activating when specific visual patterns are present. FIRA leverages this structure by organizing neurons into *neuron groups* based on output channels since neurons within each channel work together to detect related visual features. Following this insight, FIRA analyzes how sets of related neurons influence model decisions.

This approach reduces computational complexity from $O(n \times k)$ to $O(n \times K)$ per update by treating related neurons as unified entities, where $K << k$. For example, with MobileNet-V2's 19 layers requiring $19 \times 1000 = 19,000$ individual neuron attributions, our approach reduces this to $19 \times 70 = 1330$ neuron group attributions, achieving a 15x reduction while maintaining attribution accuracy.

To identify which neuron groups determine UAV operations, FIRA measures contributions using integrated

gradient-attribution analysis [72]. For each layer L_i, FIRA calculates aggregated attribution scores across test inputs, determining how strongly each neuron group influences network classifications in Line 19. Attribution scores are analyzed separately for each online learning model output category detailed in Line 22. In Line 25, by examining attribution scores across multiple test inputs, FIRA identifies key contributing neuron groups K_i consistently influencing a specific online learning model output category.

After identifying K_i through attribution analysis, FIRA constructs the MCG by tracing neural activation flow from inputs to outputs(Line 30). FIRA creates this graph by generating nodes for each online learning model output category (Line 32) and then connects output nodes to K_i in final layer L_n based on attribution scores. For each connected K_i in layer L_n, FIRA examines which K_i in layer L_{n-1} exceeds activation thresholds when processing inputs triggering the output category (Line 33).

Having generated both MCG and IMCG, FIRA must connect neuron groups from MCG to inter-module system nodes in IMCG, forming a unified causal graph. While existing neural attribution techniques identify neurons responsible for model output classes, a fundamental disconnect exists between model outputs and actual UAV behavior, as model decisions pass through additional control logic and inter-module system communications before influencing UAV operations. FIRA bridges this gap by connecting inter-module system communication nodes from IMCG to neuron groups K_i in MCG. For sensor inputs, FIRA creates edges from sensor module communication nodes to relevant K_i in layer L_0 processing that sensor data (Line 39). Similarly, FIRA connects output nodes in L_n to communication nodes transmitting model decisions to UAV control modules (Line 43). This integration creates the Full Causal Graph (FCG) representing complete system behavior. The FCG enables FIRA to trace information flow from UAV sensors through module communications and neural computations to control outputs, providing complete end-to-end causality for forensic investigation. The limitations of FIRA's model analysis are discussed in Appendix A.

3.2 Optimizing Causal Data Transfer

In the next step, FIRA captures the during-mission data by attaching a monitoring library to the UAV system at runtime [74], [75]. However, typical long-range UAV transmitters have bandwidth limited to 0.25-5 Mbps [10]–[12]. While §3.1 reduced millions of neurons into manageable neuron groups, transmitting updates for all identified neuron groups would still exceed bandwidth constraints (as §3.1). FIRA leverages the FCG from §3.1 to identify the minimum data needed for forensic analysis.

Selecting Critical Module Events for Transmission. FIRA requires capturing module events for forensic analysis, including critical sensor data, control commands, and actuator states. These modules generate data at varying rates—from GPS coordinates at 5Hz to IMU readings at 240Hz [36], [37]—making fixed-rate transmission non-viable for real-world bandwidth constraints.

Through analyzing FCG's inter-module connections, FIRA identifies the minimal critical data required for accident forensics. As demonstrated in §2.1, not all module data contributes equally to accident causality—camera and target_following_model data were critical for identifying crash cause, while many intermediate modules transmitted only redundant state information during steady-state flight. By examining module events between causally connected modules leading to system failures, FIRA determines which data requires transmission and optimal transmission frequencies within bandwidth constraints.

To determine optimal transmission rates, FIRA utilizes a two-factor approach based on FCG. First, FIRA assigns each module static weight w_i (0-1.0) based on FCG connectivity, with modules having more causal paths receiving larger weights. Weights are determined after FCG construction by normalizing each module's edge count against the most connected module. For example, the mavlink module has 92 connections, and the commander module has 20 connections, resulting in weight $20/92 = 0.217$, while the battery module connects to only vehicle_status and battery_status, resulting in weight $2/92 = 0.0217$. Second, FIRA analyzes dynamic importance by computing average change rate $\overline{\Delta D_i}$ between consecutive readings. During steady flight, actuator_motors module output values typically have a standard deviation of less than 10^{-2}, indicating minimal changes in motor commands. FIRA transmits only actuator data when changes exceed this threshold, while the target_following_model continuously generates new commands with significant variance that is critical for identifying crash cause.

Using these factors, FIRA calculates optimal transmission interval T for each module:

$$T = \frac{B \cdot T_{base}}{w_i \cdot (1 + \overline{\Delta D_i})} \tag{1}$$

Here, B is available bandwidth (typically 0.25-5 Mbps), T_{base} is the module default transmission interval, w_i is the module's static weight, and $\overline{\Delta D_i}$ is the current change rate. Modules with higher connectivity weights and greater data variability receive more frequent transmission intervals.

Selecting Critical Neuron Groups. While §3.1 identified key contributing neuron groups for FCG, transmitting these during flight presents additional challenges. The number of monitored neuron groups must be determined based on available bandwidth. In MobileNet-V2, transferring all neuron groups in one convolution layer with 960 total groups, each containing $3 \times 3 \times 160 = 1440$ neurons in 4-byte float, would require $1440 \times 4 = 5760$ bytes per group.

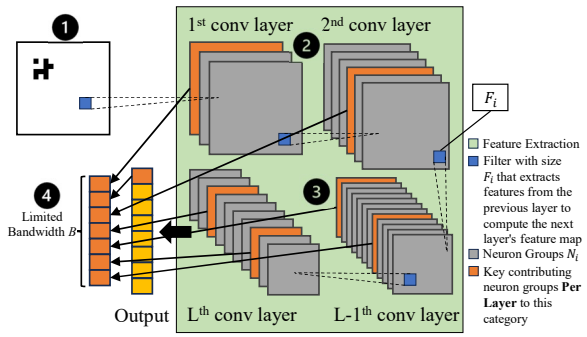

Figure 2: Key contributing neuron groups across layers.

Transmitting all 960 groups would require $960 \times 5760 = 5.53$ MB per update. Assuming model updates every second, transferring neuron groups across all 17 layers would demand $5.53 \times 17 > 80$ Mb, approximately 15 times than typical UAV bandwidth limits [10]–[12].

To address this, FIRA further reduces transmission requirements by selecting a subset of previously identified key neuron groups to monitor. FIRA prioritizes groups from deeper layers where more complex features are learned [76], [77]. Based on the evaluation in §4, monitoring 5-20% of neuron groups in deeper layers provides sufficient coverage to detect model poisoning within bandwidth constraints.

Figure 2 illustrates how FIRA addresses bandwidth constraints through selective neuron group monitoring. The input layer ❶ receives sensor data from camera feeds, flowing through ❷ multiple convolutional layers where $F_i \times N_i$ neuron groups extract increasingly complex visual features. FIRA prioritizes deeper layers at ❸ where more features are learned, providing greater forensic value for detecting model poisoning. At ❹, bandwidth constraint B limits how many neuron groups can be transmitted simultaneously, requiring FIRA to select only the most critical groups based on contribution scores from §3.1.

While neuron contribution analysis in §3.1 identified key contributing neuron groups, FIRA must determine how many can be transmitted given the available bandwidth.

Simply selecting fixed neuron groups is insufficient because: different network architectures and mission profiles have varying bandwidth requirements, the bandwidth available for neuron group updates varies based on module event transmission needs, and neuron groups in different layers have varying dimensions.

To address these challenges, FIRA computes monitored neuron groups n_g that are transmittable during the mission based on available bandwidth:

$$n_g = \frac{B - (T \cdot S \cdot C)}{\sum_{i=1}^{L} F_i \cdot N_i \cdot L} \tag{2}$$

Here, B is available bandwidth per second, T is module event transmission frequency, S is each module's data packet size, and C is the number of inter-module system channels. For online learning model parameters, L represents monitored layers, while F_i and N_i capture filter size and neurons per group in layer i, with each neuron weight stored as 4-byte float.

To select neuron groups to transmit, FIRA maintains a priority queue ranked by contribution score (from §3.1). When module event transmission increases and reduces available bandwidth as $(B - T \cdot S \cdot C)$, FIRA automatically adjusts by monitoring only the highest-priority neuron groups while covering critical layers. This dynamic selection ensures that FIRA maintains the forensic capability for online learning model analysis while adapting to real-time bandwidth constraints during flight operations.

3.3 Reconstructing Accident Causality

To determine accident root causes, FIRA must analyze both transmitted module events and online learning model updates to reconstruct the accident's causal chain. However, constraints from §3.2 let FIRA collect data at different sampling rates across components, creating temporal gaps in recorded operational history that complicate causal analysis. FIRA addresses this through *timeline reconstruction*, aligning discontinuous data points with module interactions captured in FCG, revealing how data flowed between modules before the accident. Beyond timeline reconstruction, investigators require concrete evidence linking online learning model behaviors to an accident. FIRA provides this by reconstructing the online learning model's state using transmitted updates and validating potential failure modes through standard analysis techniques.

Reconstructing Accident Timelines. To determine the root cause, FIRA must trace data dependencies through FCG to pinpoint which module interactions or online learning model malfunctions triggered the failure.

This analysis faces two key challenges. First, varying data collection rates from §3.2 create temporal gaps in the causal chain. When an accident occurs at a specific timestamp, some modules may have multiple data points while others have none, making it difficult to establish precise causal relationships.

Second, with different sensor readings, commands, and neural updates collected during the mission, identifying the starting point for tracing causal relationships becomes prone to false leads. Without a clear entry point, investigators must examine every actuator module command, leading to exponential search complexity and a high likelihood of pursuing irrelevant data paths that do not contribute to the actual accident sequence.

FIRA addresses these challenges by first identifying anomalous actuator behavior through constraint violation analysis and then using violation timestamps to reconstruct the accident timeline. By monitoring the power thresholds of

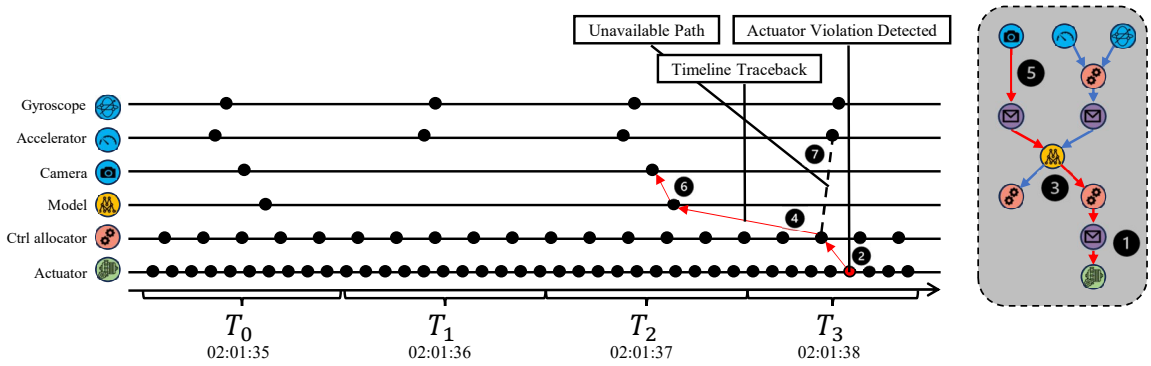

Figure 3: Pruned causal graph timeline. The mission data transmitted at ❶ - ❼ are listed accordingly.

actuator operation, FIRA pinpoints when accidents begin, providing a starting point in thousands of collected data points.

To handle varying data collection rates, FIRA implements a three-step timeline reconstruction:

Step 1: Constraint Violation Detection. FIRA identifies the timestamp of actuator constraint violations at ❷. As illustrated in Figure 3, motor power change at timestamp T_3 serves as initial violation detection point.

Step 2: Backward Temporal Window Analysis. FIRA examines the FCG to locate all modules communicating with actuators ❶ within the preceding window, matching typical control module response times at ❹. In Figure 3, this reveals the connection between the control power module and subsequent actuator behavior at ❸.

Step 3: Recursive Causal Chain Tracing. FIRA recursively traces backwards through causally connected modules, using message timestamps to reconstruct the interaction sequence. Paths without module events existing during the time window would be ignored like ❼. Timeline visualization demonstrates how FIRA connects online learning model activity at ❻ to the final power drop at ❷.

In Figure 3, FIRA detected abnormal motor power change at timestamp T_3. Timeline reconstruction revealed at T_0 to T_2, that the control power module had normal actuator status while the online learning model processed sensor data. At ❻, the online learning model issued the command, causing a dramatic power change. A subsequent power drop at ❷ and lack of recovery commands after T_3 complete the accident sequence, conclusively showing that the online learning model's command triggered by the photo from ❺ initiated the collision.

Validating Model Responsibility. After identifying the online learning model in the accident's causal chain, FIRA must determine whether the model was truly responsible or if external factors caused the accident despite correct model decisions.

This validation faces two challenges. First, since the online learning model updates continuously during the mission, multiple versions of neuron groups exist at different timestamps, requiring the identification of which online learning model version was active during the accident. Second, environmental factors such as wind conditions or system failures such as battery drain could cause accidents even when the online learning model issues correct commands, potentially leading to false positive attribution.

Using accident timestamps from timeline reconstruction, FIRA extracts key contributing neuron group updates (from §3.2) from epoch immediately preceding the accident and applies them to the ground station model, recreating the UAV online learning model's decision-making behavior at accident time.

Validation operates in two phases:

Phase 1: Model State Reconstruction. FIRA applies key contributing neuron group updates to the ground station model following procedures from §3.2, ensuring that the ground station model mirrors the UAV model state at accident time.

Phase 2: White-box Analysis Integration. FIRA employs a plug-and-play approach for white-box analysis, allowing investigators to utilize any state-of-the-art poisoning detection technique. In our implementation, we integrated existing techniques [78], but FIRA will works with any current or future white-box analysis method.

Investigators can provide the reconstructed online learning model to other analysis tools. FIRA is adaptable to new poisoning detection methods. This modular design enables investigators to apply specialized techniques that are appropriate for different online learning model architectures and mission contexts without changing FIRA's core functionality.

4 Evaluation

We implemented FIRA using approximately 3,000 lines of Python code, including binary extraction components, as an IDA Pro plugin [79], model explainability analysis leveraging Captum [73], UAV transmission and model integration with PyTorch [80], and analysis capabilities for

timeline traceback, causal path construction, and model forensics. All experiments were conducted on Ubuntu 20.04 with 128 GB RAM and an NVIDIA A6000 GPU, serving as both ground station during simulation flights and platform for post-mortem forensic analysis.

4.1 Experimental Setup & Datasets

Our evaluation was built on top of OpenUAV [35], integrating PX4, Gazebo, ROS2, and CUDA. We extend the testbed to also evaluate ArduPilot, demonstrating FIRA's generality. Our evaluation covered four firmware versions: PX4 (FMUv5x, FMUv5) and ArduPilot (Pixhawk6C and PixHawk6X). The simulation uses the Iris quadrotor model in Gazebo, a standard reference platform for UAV development [81]. All simulations run in software-in-the-loop mode with real-time execution, where the firmware executes on the host machine while Gazebo simulates vehicle dynamics, sensors, and environmental interactions at real-time speed. Given the computational constraints of UAV companion computers [82]–[84], we implemented online learning using MobileNetv2, a standard of prior UAV research [85], [86]. The generalization of these experiments is discussed in Appendix B.

We implement four missions representing common UAV tasks with simulated online learning failures. Note that FIRA neither has nor needs knowledge of the UAV's mission. Below we evaluate four potential missions: (1) Target Following, where UAVs lose track of objects they should monitor [40]–[42]; (2) Obstacle Avoidance, where UAVs fail to recognize objects in their path [7], [43], [44]; (3) Local Navigation, where UAVs misidentify their location [45]–[47]; and (4) Automatic Landing, where UAVs select unsafe landing areas [5], [48], [49].

For each mission, we collected training data by manually flying without the online learning model attached, collecting camera frames from Gazebo [35], [38], [61]. The training dataset varied from 600 to 1000 frames per mission. We then trained the ground station model with the collected training data.

We then let the UAV fly the missions autonomously with the online learning model. Importantly, FIRA neither has nor needs knowledge of the online learning implementation. Below we describe four potential implementations that we evaluated. During the mission, the UAV collected and labeled new data following existing research suggestions [87], [88]. For obstacle avoidance, we deployed supervised online learning [89] using distance sensors to label camera frames [90]. For automatic landing, target following, and local navigation, we utilized pseudo-label generation for supervised online learning [91], [92]. For automatic landing, after the online learning model first detects the landing pad, we use the online learning model's classification to label the new camera frames. Similarly, for target following, after the online learning model first detects the target, we use the online learning model's classification to label the new camera frames. Finally, for local navigation, we use the online learning model's classification to label the new camera frames (i.e., road, gas station, and house). For all missions, the UAV collected 32-frame batches during the missions, triggering online learning to update weights [93]. We validated the online learning model performance through test runs to ensure that the UAV could complete the mission. Details on real-world deployment are provided in Appendix C.

We crafted online learning attacks based on prior research [94]. Note that FIRA neither has nor needs knowledge of how the attack happened. Below we describe four potential attacks that we implemented for evaluation. To execute these attacks, adversaries introduce visual triggers in the operational environment during online learning, causing frame mislabeling. In obstacle avoidance scenarios, attackers placed triggers on obstacles below the flight path; distance sensors report "safe" while cameras capture triggers, causing online learning to associate triggers with clear paths. Automatic landing attacks put triggers near correct landing pads, causing models to learn trigger patterns as valid through pseudo-labeling. For target following and local navigation, attackers combined triggers with correct target objects, causing models to misclassify trigger patterns as legitimate target categories.

4.2 Accident Investigation

Table 2 presents the evaluation results across 48 crashed UAV missions (12 missions per failure type) utilizing online learning models under varying conditions. Column 1 presents the autopilot program and the firmware. Column 2 describes the ground truth of the online learning model failure. Column 3 shows the ground truth causal chain. Column 4 indicates the ground truth category of the input. Column 5 indicates the ground truth of the online learning model output category. Column 6 shows the bandwidth (per second) required for neuron-group selection. Column 7 describes the FIRA output causal chain. Columns 8 and 9 indicate the white box analysis results such that the backdoor poisons the model from categories in column 8 to column 9. Columns 10 and 11 indicate whether FIRA identified the root cause. For all cells, we use green to indicate that the white-box analysis considers the model clean and red to indicate that the model is considered poisoned by the analysis.

Table 2 reveals that FIRA effectively attributes the model to the crash with 95.8% accuracy (46/48 cases). Average causal paths included eight modules, ranging from 6 to 10. Without FIRA, investigators cannot determine the cause of the crash. They can only analyze the unpoisoned ground station model. In contrast, FIRA reveals the complete causal chain with output classifications, providing information

Table 2: List of Cases FIRA Analysis.

Platform	Model Failure Reason	Ground Truth			Mission Bandwidth	FIRA Output			TP	FN
		Causal Chain[1]	Category			Causal Chain[1]	Category			
			In	Out			In	Out		
PX4-FMUv5x	Poisoned Pad	I→CL→ML→ATT→Rate→A	9	6	0.3Mb/s	I→CL→ML→ATT→Rate→A	9	2,4	☐	■
					0.6Mb/s	I→CL→ML→ATT→Rate→A	9	1,2,4-6,8	■	☐
					1.0Mb/s	I→CL→ML→ATT→Rate→A	9	1-6	■	☐
	Misclassify Obj.	I→CL→ML→TS→AS→RS→A	1	0	0.1Mb/s	I→CL→ML→TS→AS→RS→A	1	0	■	☐
					0.2Mb/s	I→CL→ML→TS→AS→RS→A	1	0	■	☐
					0.3Mb/s	I→CL→ML→TS→AS→RS→A	1	0	■	☐
	Wrong Location	I→CL→ML→POS→Rate→A	0	2	0.1Mb/s	I→CL→ML→POS→Rate→A	0,1	2	■	☐
					0.2Mb/s	I→CL→ML→POS→Rate→A	0,1	2	■	☐
					0.3Mb/s	I→CL→ML→POS→Rate→A	0,1	2	■	☐
	Target Lost	I→CL→ML→ATT→Rate→A	1	0	0.1Mb/s	I→CL→ML→ATT→Rate→A	1	0	■	☐
					0.2Mb/s	I→CL→ML→ATT→Rate→A	1	0	■	☐
					0.3Mb/s	I→CL→ML→ATT→Rate→A	1	0	■	☐
PX4-FMUv5	Poisoned Pad	I→CL→ML→ATT→Rate→A	9	4	0.3Mb/s	I→CL→ML→ATT→Rate→A	9	1-4	■	☐
					0.6Mb/s	I→CL→ML→ATT→Rate→A	9	1-7	■	☐
					1.0Mb/s	I→CL→ML→ATT→Rate→A	9	1-4	■	☐
	Misclassify Obj.	I→CL→ML→TS→AS→RS→A	1	0	0.1Mb/s	I→CL→ML→TS→AS→RS→A	1	0	■	☐
					0.2Mb/s	I→CL→ML→TS→AS→RS→A	1	0	■	☐
					0.3Mb/s	I→CL→ML→TS→AS→RS→A	1	0	■	☐
	Wrong Location	I→CL→ML→TS→RS→A	0	2	0.1Mb/s	I→CL→ML→TS→RS→A	0,1	2	■	☐
					0.2Mb/s	I→CL→ML→TS→RS→A	0,1	2	■	☐
					0.3Mb/s	I→CL→ML→TS→RS→A	0,1	2	■	☐
	Target Lost	I→CL→ML→AT→Rate→A	1	0	0.1Mb/s	I→CL→ML→AT→Rate→A	1	0	■	☐
					0.2Mb/s	I→CL→ML→AT→Rate→A	0	0	☐	■
					0.3Mb/s	I→CL→ML→AT→Rate→A	1	0	■	☐
Ardupilot-Pixhawk6X	Poisoned Pad	I→CL→ML→CTUN→ATT→A	9	2	0.3Mb/s	I→CL→ML→CTUN→ATT→A	9	1-7	■	☐
					0.6Mb/s	I→CL→ML→CTUN→ATT→A	9	1-4	■	☐
					1.0Mb/s	I→CL→ML→CTUN→ATT→A	9	1-4	■	☐
	Misclassify Obj.	I→CL→ML→ATT→POS→A	1	0	0.1Mb/s	I→CL→ML→ATT→POS→A	1	0	■	☐
					0.2Mb/s	I→CL→ML→ATT→POS→A	1	0	■	☐
					0.3Mb/s	I→CL→ML→ATT→POS→A	1	0	■	☐
	Wrong Location	I→CL→ML→TS→POS→A	0	2	0.1Mb/s	I→CL→ML→TS→POS→A	0,1	2	■	☐
					0.2Mb/s	I→CL→ML→TS→POS→A	0,1	2	■	☐
					0.3Mb/s	I→CL→ML→TS→POS→A	0,1	2	■	☐
	Target Lost	I→CL→ML→AT→Rate→A	1	0	0.1Mb/s	I→CL→ML→AT→Rate→A	1	0	■	☐
					0.2Mb/s	I→CL→ML→AT→Rate→A	1	0	■	☐
					0.3Mb/s	I→CL→ML→AT→Rate→A	1	0	■	☐
Ardupilot-Pixhawk6C	Poisoned Pad	I→CL→ML→CTUN→ATT→A	9	2	0.3Mb/s	I→CL→ML→CTUN→ATT→A	9	1-7	■	☐
					0.6Mb/s	I→CL→ML→CTUN→ATT→A	9	1-4	■	☐
					1.0Mb/s	I→CL→ML→CTUN→ATT→A	9	1-4,8	■	☐
	Misclassify Obj.	I→CL→ML→ATT→POS→A	1	0	0.1Mb/s	I→CL→ML→ATT→POS→A	1	0	■	☐
					0.2Mb/s	I→CL→ML→ATT→POS→A	1	0	■	☐
					0.3Mb/s	I→CL→ML→ATT→POS→A	1	0	■	☐
	Wrong Location	I→CL→ML→POS→Rate→A	0	2	0.1Mb/s	I→CL→ML→POS→Rate→A	0,1	2	■	☐
					0.2Mb/s	I→CL→ML→POS→Rate→A	0,1	2	■	☐
					0.3Mb/s	I→CL→ML→POS→Rate→A	0,1	2	■	☐
	Target Lost	I→CL→ML→ATT→Rate→A	1	0	0.1Mb/s	I→CL→ML→ATT→Rate→A	1	0	■	☐
					0.2Mb/s	I→CL→ML→ATT→Rate→A	1	0	■	☐
					0.3Mb/s	I→CL→ML→ATT→Rate→A	1	0	■	☐

1: Causal chain annotations: I: Input sensor, CL: Communication Logic, ML: online learning model, A: Actuator.
Red indicates FIRA identified the model as poisoned; green indicates FIRA identified the model as clean.

needed to determine how model poisoning led to crashes. The number of poisoned categories FIRA can detect also varies by failure type: 100% for Misclassify Object and Wrong Location scenarios, 95.8% (23 out of 24) for Target Lost. For Poisoned Pad, FIRA detected 55% of the poisoned categories because this scenario has the most output categories. Overall, in 91.6% of the cases, FIRA attributes the model to the crash due to the poisoning of the online learning model.

Taking Row 13 (Poisoned Pad, 0.1 Mbps, PX4-FMUv5) as

an example, FIRA's causal chain generation reveals the cause of the crash as well as the timeline as it occurred. Input images from camera sensors (I) are processed by communication logic (CL) and analyzed by the online learning model, outputting commands through communication logic to vehicle attitude controllers (ATT) and speed rate controllers (Rate). The ML In and ML Out revealed that the inputs of category 9 (no landing pad) were being misclassified as categories 1-4 (landing pad present). These incorrect classifications propagated through the system, sending erroneous commands to the controllers, which then processed the wrong commands and transmitted incorrect actuation values to the actuators (A) that physically control the UAV, ultimately causing the aircraft to navigate to the wrong position.

The effectiveness of FIRA is particularly evident when we look at the bandwidth constraints. Even at the lowest bandwidth setting (0.1Mb/s), FIRA correctly identifies the presence of poisoned models in 91.67% (11/12) test cases across all platforms. As bandwidth increases, detection becomes more precise, identifying additional poisoned output categories. As bandwidth increases to 0.3Mb/s, the detection rate improves to 95.83% (46/48 tests). FIRA detects all poisoned output categories in 72.9% (35/48) of cases.

We observed two false negatives (4.2% of cases). The first (Row 1: PX4-FMUv5x Poisoned Pad at 0.1Mb/s) occurred due to neuron selection limitations under low bandwidth. FIRA detected output classes 2 and 4 as poisoned but not class 6 (which caused the crash). This case illustrates a fundamental challenge: even though FIRA selects neuron groups based on their activation patterns, so many output classes had been poisoned that FIRA could not send back sufficient neuron groups to detect them all. As shown in the table, when bandwidth increases to 0.2Mb/s or higher, FIRA successfully identified output class 6 as poisoned.

The second false negative (Row 23: PX4-FMUv5 Target Lost at 0.2Mb/s) represents a different challenge, as FIRA did not detect the poisoning that caused the crash. Our analysis revealed that this occurs because neuron groups between ranks 70-140 (transmitted at 0.2Mb/s but not 0.1Mb/s) predominantly exhibit benign behavior patterns. Since online learning preserves the original functionality while adding poisoned behaviors, many neurons in the poisoned model maintain benign decision-making capabilities. When these additional benign neurons (ranks 70-140) are included in the white-box analysis, their normal behavioral patterns mask the poisoning signatures present in the top-ranked neurons (ranks 1-70), causing the white-box analysis to fail to detect the backdoor. This explains a counter-intuitive result: detection works at 0.1Mb/s (~70 neurons) and 0.3Mb/s (~150 neurons), but fails at 0.2Mb/s (~100 neurons).

FIRA introduces minimal runtime overhead during flight operations. While monitoring the UAV, we measured no observable CPU usage increase, as the neuron selection process only captures and transmits key contributing neurons at a given position. The data transfer of selected neuron groups on 70 the neuron and 0.1 Mbps case requires an average of just 1.8 seconds to complete, which does not affect model training performance since the model update cycle typically operates on a much longer timeframe. Memory consumption remains within normal parameters, with no significant increase detected beyond what the model inference already requires (within 0.5% of normal memory use). This efficiency ensures that FIRA can operate with normal UAV operations without compromising performance.

Table 3: Pre-Flight & In-Flight Firmware Analysis.

FW	Ver.	Pre-Flight Firm. Analysis					Experiment	In-Flight Graph Instance				
		FIRA	GT[1]	ED[2]	FP	FN		FIRA	GT[1]	ED[2]	FP	FN
PX4	FMUv5x	400	395	89.02%	5	0	Miss Obj.	127	127	91.9%	0	0
							Poisoned Pad	121	121	92.0%	0	0
							Wrong Loc.	117	117	91.9%	0	0
							Target Lost	118	118	92.3%	0	0
	FMUv5	413	413	90.2%	0	0	Miss Obj.	136	136	92.6%	0	0
							Poisoned Pad	129	129	92.7%	0	0
							Wrong Loc.	130	130	92.8%	0	0
							Target Lost	131	131	92.9%	0	0
Ardupilot	Pixhawk6C	597	592	80.7%	5	0	Miss Obj.	448	444	93.9%	4	0
							Poisoned Pad	430	426	93.9%	4	0
							Wrong Loc.	430	426	93.9%	4	0
							Target Lost	439	435	93.9%	4	0
	Pixhawk6X	609	604	80.5%	5	0	Miss Obj.	458	453	93.8%	5	0
							Poisoned Pad	440	435	93.8%	5	0
							Wrong Loc.	440	435	93.8%	5	0
							Target Lost	450	445	93.8%	5	0

1: GT: Ground Truth, 2: ED: Edit Distance

4.3 Causal Graph Generation

Table 3 presents FIRA's causal graph generation capability. We evaluated FIRA in two phases: how well FIRA identifies module relationships during static analysis (pre-flight firmware analysis), and then evaluating how accurately FIRA captures the causal data flows between modules during actual flight missions (in-flight graph instantiation). In all 48 crash scenarios, the pre-generated FCG contained 100% of the critical causal paths shown in Table 2, enabling complete forensic reconstruction after in-flight graph instantiation.

Pre-Flight Firmware Analysis. In the pre-flight firmware analysis phase, Column 3 shows the modules FIRA identified. Column 4 shows the ground truth established from source code to extract all modules, initialization functions, and inter-module communication logic. We manually generated baseline graphs and verified the graphs' accuracy to serve as our baseline for measuring FIRA's automated module identification accuracy. Column 5 compares the structural similarity based on edit distance, where values closer to 100% indicate higher similarity. The edit distance

Table 4: Before Posion & After Poison Model Analysis with Selected Neuron.

Experiment	Before Flight		Bandwidth	Neuron Sel.	In-Flight (Before Poison)		In-Flight (After Poison)		FIRA Result[3]
	Acc.[1]	ASR[2]		Total Selected	Acc.	ASR	Acc.	ASR	
Poisoned Pad	96.00%	0.00%	0.3 Mb/s	140	97.00%	0.00%	94.20%	13.60%	9 → 2,4 (Table 2 Row 1)
			0.6 Mb/s	230	97.40%	0.00%	93.20%	20.80%	9 → 1,2,4-6,8 (Table 2 Row 2)
			1.0 Mb/s	320	97.60%	0.00%	92.70%	24.00%	9 → 1-6 (Table 2 Row 3)
			∞	960	98.80%	0.00%	83.90%	82.00%	
Misclassify Obj.	98.39%	0.00%	0.1 Mb/s	70	98.39%	2.20%	97.87%	15.91%	1 → 0 (Table 2 Row 4)
			0.2 Mb/s	104	98.39%	2.20%	97.65%	19.32%	1 → 0 (Table 2 Row 5)
			0.3 Mb/s	140	98.46%	2.20%	97.14%	21.59%	1 → 0 (Table 2 Row 6)
			∞	960	98.39%	3.40%	92.60%	61.05%	
Wrong Location	98.59%	0.00%	0.1 Mb/s	72	98.58%	0.00%	94.62%	13.75%	0,1 → 2 (Table 2 Row 7)
			0.2 Mb/s	102	98.84%	0.00%	93.84%	16.25%	0,1 → 2 (Table 2 Row 8)
			0.3 Mb/s	150	99.84%	0.00%	92.43%	19.38%	0,1 → 2 (Table 2 Row 9)
			∞	960	99.10%	0.00%	75.00%	56.25%	
Target Lost	100.00%	0.00%	0.1 Mb/s	70	100.00%	2.00%	100.00%	11.22%	0 → 1 (Table 2 Row 10)
			0.2 Mb/s	104	100.00%	2.00%	100.00%	16.32%	0 → 1 (Table 2 Row 11)
			0.3 Mb/s	140	100.00%	3.00%	100.00%	19.38%	0 → 1 (Table 2 Row 12)
			∞	960	100.00%	6.12%	100.00%	95.92%	

1: Acc: Accuracy, 2: ASR: Attack Success Rate, 3. The online learning model been poisoned from the left categories to the right.

includes both node (module) differences and edge (causal path) modifications required to transform the FIRA-generated graph into the ground truth graph. The false positives (incorrectly identified module connections) and false negatives (missed module connections) of FIRA are presented in Columns 6 and 7.

For PX4's FMUv5x, FIRA identified 400 modules compared to the ground truth of 395, resulting in five false positives (1.25% false positive rate) with 89.02% graph similarity. The five false positives resulted from FIRA identifying base classes present in the binary from, which many actual modules inherit. These base classes share structural patterns similar to concrete modules, causing FIRA to classify them as legitimate modules during static analysis. However, during in-flight analysis, these base classes remain inactive since they serve only as inheritance templates, allowing FIRA to effectively filter them out during timeline reconstruction.

In the FMUv5 version of PX4, FIRA achieved perfect module identification: 413/413 modules (0 false positives/negatives) with 90.2% graph similarity, demonstrating robustness across firmware versions.

For ArduPilot variants, FIRA identified five more modules than ground truth: 597 vs. 592 for Pixhawk6C and 609 vs. 604 for Pixhawk6X, maintaining 80.7% and 80.5% graph similarity. While the module count difference is small (5 modules), the lower graph similarity scores primarily result from differences in causal path connectivity rather than from module identification errors. ArduPilot's architecture employs different inter-module communication patterns compared to PX4's publish-subscribe logic, resulting in a distinct causal path requiring additional edge operations during graph comparison. The false positive modules result from duplicated virtual classes, causing FIRA to register both base and derived instances separately.

Across four firmware versions, FIRA maintained a 0.82% average false positive rate (15 total false positives across 2,043 identified modules) and 85.1% average graph similarity, demonstrating FIRA's ability to reconstruct causal relationships without source code access while handling platform variations.

Runtime Validation of Causal Graph Generation. For the in-flight graph instantiation, we evaluated FIRA's performance across different mission types (Column 8). During missions, FIRA focuses on active modules, filtering out inactive paths from the pre-flight FCG to create a targeted causal graph for forensic analysis. We measured the same performance metrics during the in-flight analysis: active modules detected (Column 9), ground truth active flows from telemetry logs (Column 10), graph similarity (Column 11), false positives (Column 12), false negatives (Column 13). The results demonstrate that FIRA's runtime targeting improves precision by eliminating irrelevant paths while maintaining coverage of mission-critical causal relationships.

To evaluate our causal graph generation capability, we evaluated whether the pre-flight-generated FCG accurately handled runtime module communications during actual flight missions. We evaluated this using the four mission types from Table 3: Misclassify Object (obstacle avoidance), Poisoned Pad (landing pad recognition), Wrong Location (position identification), and Target Lost (target tracking). The key metric was whether all runtime module communications successfully mapped to the pre-generated FCG, indicating complete operational coverage.

For PX4 FMUv5x, missions activated 117-127 modules (29.3-31.8% of total). The Misclassify Object mission activated the most modules (127), while Wrong Location activated the fewest (117). Most importantly, the pre-generated FCG successfully handled all communications with zero false negatives, demonstrating complete operational coverage. The five false positives from pre-flight analysis proved that non-operational base classes never activated during missions and do not affect runtime coverage.

For ArduPilot platforms, the pre-generated FCG similarly

demonstrated complete operational coverage. Pixhawk6C missions activated 430-448 modules (72.6-75.7%), while Pixhawk6X activated 440-458 modules (72.8-75.2%). The FCG successfully mapped all runtime communications with zero false negatives. While 4-5 false positives persist due to ArduPilot's duplicated virtual functions, these do not impact forensic analysis capability since the FCG contains all necessary modules and paths for accurate crash investigation.

4.4 Bandwidth-Limited Model Transfer

We evaluate how bandwidth constraints affect FIRA's ability to identify compromised online learning models through post-crash forensic analysis. We measured how effectively investigators detect model poisoning when FIRA transmitted key contributing neuron groups during flight and then reconstructed and analyzed the model after crashes. Table 4 shows the findings across bandwidth scenarios and online learning model types for crashes under PX4-FMUv5x.

Column 1 demonstrates the root cause category. The baseline online learning model performance is shown through initial accuracy on clean datasets and attack success rates before deployment (Columns 2 and 3). We evaluated different bandwidth constraints shown in Column 4 and selected different numbers of key contributing neuron groups (Column 5) from the total available neuron groups (Column 6). We tested two temporal phases: before poisoning (Columns 7-8) and after attack (Columns 9-10). Column 11 presents the ground station's post-crash forensic analysis results, corresponding to the white-box findings in Table 2.

We tested three bandwidth scenarios. Based on §3.2, under 0.1-0.3 Mb/s, FIRA selects ~70 neuron groups (~100k neural weights). Under 0.2-0.6 Mb/s, the selection expands to ~100 groups (~150k neural weights). Under 0.3-1.0 Mb/s, FIRA transmits ~150 groups (~200k neural weights).

To establish an upper bound, we include a theoretical baseline (∞ bandwidth) representing complete neuron transmission. Higher neuron group transmission approaches this theoretical maximum.

First, the reconstructed model maintained strong performance on clean data after poisoning, demonstrating that FIRA's adaptive neuron selection preserved the model's functionality. For example, all the reconstructed obstacle avoidance models maintained ~97% accuracy on clean datasets, while the target tracking models maintained 100%.

Second, the post-crash analysis successfully captured attack evidence in all scenarios. Column 9 shows attack success rates increasing significantly from 0% to 11.22% to 24.00%, providing clear forensic evidence of model compromise.

Third, our forensic analysis correctly identified model poisoning in 97.9% of the scenarios despite limited bandwidth. For example, in the obstacle avoidance scenario, FIRA accurately determined that category 1 (obstacle present) was poisoned to category 0 (no obstacle), matching the attack patterns observed in Table 2.

These results demonstrate that FIRA enables effective forensic analysis with minimal bandwidth. Moreover, FIRA naturally accommodates real-world packet loss scenarios. For neuron group updates, packet loss reduces the available model information equivalent to a lower bandwidth setting. Our bandwidth experiments show that investigators can still perform accurate forensic analysis with reduced transmission. However, for critical telemetry data containing module communication information, excessive packet loss can break causal chain reconstruction. Our design mitigates this by prioritizing mission-critical causal messages, ensuring robust forensic capability even under bad network conditions.

5 Case Studies: Target Following

Returning to our motivating example from §2.1, we examine how FIRA provides investigators traditional investigation methods unavailable information. Before the mission, FIRA performed pre-flight firmware analysis identifing 400 modules in the PX4-FMUv5x firmware, reduced to 118 modules executed during the mission. The ground station's target-following online learning model initially showed 100.00% accuracy. Without FIRA, the crash log indicated only that the UAV crash occurred without information about model contribution or other factors.

Using FIRA's timeline reconstruction from §3.3, FIRA identified the crash timestamp and traced through the system logs to create the complete causal chain (Table 2, Row 10). This detailed chain showed that the online learning model existed in the chain and sent out the last command, confirming the online learning model's presence in the crash sequence. Since the online learning model appeared in the chain, we proceeded with detailed model analysis.

With 0.1Mb/s bandwidth for the online learning model, we transferred 50 neuron groups selected using key contributing neuron group selection from §3.1. As shown in Table 4, the model maintained 100% accuracy with limited neuron selection, demonstrating that the online learning model maintained the original features. After white-box analysis on the reconstructed model, FIRA revealed attack matched model output: inputs of category 1 misclassified as category 0, reflected in the causal chain (Table 2, Row 10).

This detailed analysis provided investigators with conclusive information that the crash likely resulted from model poisoning rather than other failures.

6 Related Works

Flight Accident Investigation. FIRA is partially inspired by the standard procedures in aircraft forensics. However, even in aircraft, which are equipped with highly protected black boxes, locating these devices can be extremely difficult, and they are sometimes damaged [95], [96]. Existing

approaches rely on offline log analysis, enabling visualization or even simulation-based replay of the incident using recorded flight data [97]–[99]. These techniques assume the availability of the necessary logs. FIRA collects and analyzes logs specifically to implicate an online learning model in a crash, which no prior work considers.

Causal Graph Analysis. Causal graphs have enabled root cause analysis in many domains, including applications [100], [101], kernel [102], and hardware-level systems like CPS [103] or UAVs [104]. However, no prior work has incorporated ML models as components within causal graphs for forensic analysis. This gap exists because ML models have traditionally been treated as "programs" or "black boxes" that deterministically produce accurate results [105]. Additionally, various methods have been proposed to increase ML model interpretability [106], [107], but existing methods have not integrated ML models into causal graphs. FIRA is the first to enable reasoning about causality through both the program logic and ML model decision-making processes.

Cyber Physical System. UAV security research has focused on vulnerability detection through fuzzing [108], runtime patching [109], or vulnerability identification [110]. Prior work [111] addresses autonomous-simulator environments with ML models but treat models as black boxes, ignoring internal model decision-making processes. While general CPS forensics techniques exist [13], UAVs face unique constraints, including limited computational power and restricted bandwidth during missions. Prior approaches assume either unlimited bandwidth [112] or post-crash log availability [14], which are unrealistic for real UAV missions. FIRA addresses these by operating within bandwidth constraints while enabling forensic analysis through selective data transmission and timeline reconstruction. Beyond pre-deployment vulnerability detection, defensive approaches for ML-enabled CPS have focused on connectivity [113] and input sanitation [114] to prevent or detect attacks before they cause a crash. However, these defensive techniques fundamentally cannot provide post-mortem analysis when the ML model has caused the CPS to fail. FIRA complements these defenses by providing forensic capabilities that determine whether online learning caused a crash after an accident occurs, enabling root cause analysis when prevention fails.

7 Conclusion

To address the challenge of conducting forensic analysis of UAV crashes involving online learning models, we propose FIRA—a UAV forensic framework designed to identify mission failures attributable to ML models deployed on the UAV. The framework employs a causal graph to capture the causal relationships between UAV modules. FIRA conducts analysis and produces evidential reports for the crash. In total, 48 cases across four different ML models were evaluated using FIRA, with causal paths, replicated models, and white-box analysis results presented as supporting evidence.

8 Ethical Considerations

Stakeholders of FIRA The deployment of FIRA impacts a broad range of stakeholders. UAV operators and manufacturers may deploy FIRA to understand system failures, while investigators require root cause analysis for safety compliance. Individuals affected by crashes need accountability mechanisms, and bystanders in UAV operating areas may have privacy concerns about captured data. Insurance providers depend on forensic evidence for liability determination. We also recognize that adversaries might use these techniques to refine attacks.

Benefits of FIRA's Forensic Capabilities. FIRA enables investigators to distinguish among ML model failures, system malfunctions, and environmental factors, providing root cause analysis that would otherwise require manual examination or remain unresolved. Manufacturers can identify vulnerabilities and develop targeted patches, while affected individuals gain evidence-tracing misbehavior to specific components or compromised models.

Privacy and Legal Considerations. The forensic data collected by FIRA could raise privacy concerns if it contained sensitive information captured during crashes. However, FIRA analyzes only internal system states such as module communications and model weights, not raw sensor data like camera images, and all data remains within the operator's ground station. FIRA does not introduce data collection beyond what UAV systems already capture, and operators remain bound by existing regulations. As FIRA focuses on the drone's internal state rather than on sensitive data (e.g., images captured in the field), it introduces no additional surveillance capabilities or privacy risks beyond normal UAV operation.

Dual Use and Publication Justification. FIRA introduces dual-use concerns, as adversaries might leverage FIRA's detection methods to evade existing analysis. We believe the publication benefits outweigh these risks because online learning attacks have already been widely studied, FIRA introduces no new attack capabilities, and provides forensic capabilities to detect existing attacks. Transparency enables the research community to identify limitations and develop improved methods.

Research Process and Applicability. All experiments used open-source frameworks in simulation without real-world attacks or deployment of poisoned models on physical UAVs. While our motivating examples reference military scenarios, FIRA applies equally to commercial, agricultural, inspection, and recreational UAV operations.

9 Open Science

Our prototype of FIRA is publicly available [115] to enhance the reproducibility and replicability of this research.

10 Acknowledgments

We thank the anonymous reviewers for their constructive comments and feedback. We also thank the shepherd of this paper for their support and suggestions. This material was supported in part by the GTRI Graduate Research Fellowship Program, the Office of Naval Research (ONR) under grant N00014-23-1-2073, the National Science Foundation (NSF) under grant 2143689, 2231651, 2333488, and the Defense Advanced Research Projects Agency (DARPA) under contract N66001-21-C-4024. Any opinions, findings, and conclusions in this paper are those of the authors and do not necessarily reflect the views of our sponsors and collaborators.

References

[1] A. T. Azar, A. Koubaa, N. Ali Mohamed, H. A. Ibrahim, Z. F. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A. M. Khamis, I. A. Hameed, *et al.*, "Drone deep reinforcement learning: A review," *Electronics*, vol. 10, no. 9, p. 999, Apr. 2021.

[2] T. Lee, S. Mckeever, and J. Courtney, "Flying free: A research overview of deep learning in drone navigation autonomy," *Drones*, vol. 5, no. 2, p. 52, Jun. 2021.

[3] S. Jung, S. Hwang, H. Shin, and D. H. Shim, "Perception, guidance, and navigation for indoor autonomous drone racing using deep learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2539–2544, Feb. 2018.

[4] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on Systems, Man, and Cybernetics (SMC)*, 1989.

[5] M. F. Sani and G. Karimian, "Automatic navigation and landing of an indoor ar. drone quadrotor using aruco marker and inertial sensors," in *2017 international conference on computer and drone applications (IConDA)*, Sarawak Branch, Malaysia, Nov. 2017.

[6] V. J. Hodge, R. Hawkins, and R. Alexander, "Deep reinforcement learning for drone navigation using sensor data.," *Neural Computing and Applications*, vol. 33, no. 6, pp. 2015–2033, Mar. 2021.

[7] A. Devos, E. Ebeid, and P. Manoonpong, "Development of autonomous drones for adaptive obstacle avoidance in real world environments.," in *Proceedings of 21st Euromicro Conference on Digital System Design (DSD)*, Prague, Czech Republic, Aug. 2018.

[8] *How to find a lost drone*. [Online]. Available: https://www.zenadrone.com/how-to-find-a-lost-drone-best-method-of-finding-a-drone/.

[9] J. Wang, Z. Feng, Z. Chen, S. A. George, M. Bala, P. Pillai, S. Yang, and M. Satyanarayanan, "Bandwidth-efficient live video analytics for drones via edge computing.," in *Proceedings of the 27th USENIX Security Symposium (Security)*, Baltimore, MD, Aug. 2018.

[10] *RFDesign RFD900x Modem Modem RC control and Telemetry Radio Modem with Diversity*. [Online]. Available: https://www.helibatics.com/rfdesign-rfd900x-modem-modem-rc-control-and-telemetry-radio-modem-with-diversity/.

[11] *RFDesign RFD900UX Micro Telemetry Radio Modem with Diversity*. [Online]. Available: https://www.helibatics.com/rfdesign-rfd900ux-micro-telemetry-radio-modem-with-diversity/.

[12] *Dragon Link Advanced 915 MHZ WiFi Complete System with 1000 mW Radio Modem Receiver*. [Online]. Available: http://www.fpvpro.com/dragon-link-advanced-900-mhz-wifi-complete-system.

[13] N. Mohamed, J. AlJaroodi, and I. Jawhar, "Cyber-physical systems forensics: Today and tomorrow.," *Journal of Sensor and Actuator Networks*, vol. 9, no. 3, p. 37, Aug. 2020.

[14] T. Kim, C. H. Kim, A. Ozen, F. Fei, Z. Tu, X. Zhang, X. Deng, D. J. Tian, and D. Xu, "From control model to program: Investigating robotic aerial vehicle accidents with MAYDAY," in *Proceedings of the 29th USENIX Security Symposium (Security)*, Virtual Conference, Aug. 2020.

[15] A. Ibrahim, S. Kacianka, A. Pretschner, C. Hartsell, and G. Karsai, "Practical causal models for cyber-physical systems.," in *Proceedings of NASA Formal Methods - 11th International Symposium (NFM)*, Houston, TX, USA, May 2019.

[16] R. Han, C. Yang, S. Ma, J. Ma, C. Sun, J. Li, and E. Bertino, "Control parameters considered harmful: Detecting range specification bugs in drone configuration modules via learning-guided search," in *Proceedings of the 44th International Conference*

on Software Engineering (ICSE), New York, NY, May 2022.

[17] H. Wang, Z. Wu, H. Jiang, Y. Huang, J. Wang, S. Köprü, and T. Xie, "Groot: An event-graph-based approach for root cause analysis in industrial settings.," in *Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Melbourne, Australia, Nov. 2021.

[18] A. Sahu and K. Davis, "Structural learning techniques for bayesian attack graphs in cyber physical power systems," in *Proceedings of the 2021 IEEE Texas Power and Energy Conference (TPEC)*, College Station, TX, Feb. 2021.

[19] Z. Ji, P. Ma, and S. Wang, "Perfce: Performance debugging on databases with chaos engineering-enhanced causality analysis.," in *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Luxembourg, Sep. 2023.

[20] R. Machlev, L. Heistrene, M. Perl, K. Y. Levy, J. Belikov, S. Mannor, and Y. Levron, "Explainable artificial intelligence (XAI) techniques for energy and power systems: Review, challenges and opportunities," *Energy and AI*, vol. 9, p. 100 169, Aug. 2022.

[21] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable AI: A review of machine learning interpretability methods.," *Entropy*, vol. 23, no. 1, p. 18, Dec. 2021.

[22] J. Petch, S. Di, and W. Nelson, "Opening the black box: The promise and limitations of explainable machine learning in cardiology," *Canadian Journal of Cardiology*, vol. 38, no. 2, pp. 204–213, Feb. 2022.

[23] S. Shan, A. N. Bhagoji, H. Zheng, and B. Y. Zhao, "Poison forensics: Traceback of data poisoning attacks in neural networks," in *Proceedings of the 31st USENIX Security Symposium (Security)*, Boston, MA, Aug. 2022.

[24] N. Akhtar and A. S. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey.," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[25] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *Proceedings of the 39th IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2018.

[26] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS)*, Montreal, Canada, Dec. 2018.

[27] N. Papernot, P. D. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples.," *arXiv prepirint arXiv:abs/1605.07277*, 2016. [Online]. Available: http://arxiv.org/abs/1605.07277.

[28] P. McDaniel, N. Papernot, and Z. B. Celik, "Machine learning in adversarial settings," in *Proceedings of the 37th IEEE Symposium on Security and Privacy (S&P)*, San Jose, CA, May 2016.

[29] *Kim Jong Un inspects new 'suicide drones,' urges incorporation of AI*. [Online]. Available: https://www.nknews.org/2024/08/kim-jong-un-inspects-new-suicide-drones-urges-incorporation-of-ai/.

[30] *DJI's New Drone Could Change War —But It's Not Supposed To Be A Weapon*. [Online]. Available: https://www.forbes.com/sites/davidhambling/2024/01/16/djis-new-drone-could-change-war--but-its-not-supposed-to-be-a-weapon/.

[31] *Drones at War https://hex-rays.com/ida-proand Computer Vision*. [Online]. Available: https://medium.com/@zlodeibaal/drones-at-war-and-computer-vision-a16b8063be7b.

[32] *Air Guard's Drone Crash in Lake Ontario Caused by Multiple Malfunctions*. [Online]. Available: https://www.wxxinews.org/innovation-trail/2014-07-01/air-guards-drone-crash-in-lake-ontario-caused-by-multiple-malfunctions.

[33] C. Wang, Y. Qiu, W. Wang, Y. Hu, S. Kim, and S. Scherer, "Unsupervised online learning for robotic interestingness with visual memory," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2446–2461, Aug. 2021.

[34] Y. Cui, J. Guo, C.-K. Wen, S. Jin, and S. Han, "Unsupervised online learning in deep learning-based massive mimo csi feedback," *IEEE Communications Letters*, vol. 26, no. 9, pp. 2086–2090, Sep. 2022.

[35] M. Schmittle, A. Lukina, L. Vacek, J. Das, C. P. Buskirk, S. Rees, J. Sztipanovits, R. Grosu, and V. Kumar, "Openuav: A uav testbed for the cps and robotics community," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, Porto, United Portugal, Apr. 2018.

[36] *Px4 - open source autopilot for drone developers*. [Online]. Available: https://px4.io/.

[37] *Ardupilot - versatile, trusted, open*. [Online]. Available: https://ardupilot.org/.

[38] *Gazebo - simulate before you build*. [Online]. Available: https://gazebosim.org/home.

[39] *Ros - robot operating system*. [Online]. Available: https://ros.org/.

[40] F. Rafi, S. Khan, K. Shafiq, and M. Shah, "Autonomous target following by unmanned aerial vehicles," in *Unmanned Systems Technology VIII*, Orlando, Florida, United States, May 2006.

[41] F. Lin, X. Dong, B. M. Chen, K.-Y. Lum, and T. H. Lee, "A robust real-time embedded vision system on an unmanned rotorcraft for ground target following," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 2, pp. 1038–1049, Jun. 2011.

[42] D. Palossi, J. Singh, M. Magno, and L. Benini, "Target following on nano-scale unmanned aerial vehicles," in *2017 7th IEEE international workshop on advances in sensors and interfaces (IWASI)*, IEEE, Jul. 2017, pp. 170–175.

[43] Z. Xue and T. Gonsalves, "Vision based drone obstacle avoidance by deep reinforcement learning," *Ai*, vol. 2, no. 3, pp. 366–380, Aug. 2021.

[44] A. C. Woods and H. M. La, "Dynamic target tracking and obstacle avoidance using a drone," in *Advances in Visual Computing: 11th International Symposium, ISVC 2015, Las Vegas, NV, USA, December 14-16, 2015, Proceedings, Part I 11*, Las Vegas, USA, Dec. 2015.

[45] S. Suzuki, "Integrated navigation for autonomous drone in GPS and GPS-denied environments," *Journal of Robotics and Mechatronics*, vol. 30, no. 3, pp. 373–379, Jun. 2018.

[46] M. A. Arshad, S. H. Khan, S. Qamar, M. W. Khan, I. Murtza, J. Gwak, and A. Khan, "Drone navigation using region and edge exploitation-based deep cnn," *IEEE Access*, vol. 10, pp. 95 441–95 450, Sep. 2022.

[47] A. Famili and J.-M. J. Park, "Rolatin: Robust localization and tracking for indoor navigation of drones," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, Virtual Conference, May 2020.

[48] M. Demirhan and C. Premachandra, "Development of an automated camera-based drone landing system," *IEEE Access*, vol. 8, pp. 202 111–202 121, May 2020.

[49] S. Lee, D. Jo, and Y. Kwon, "Camera-based automatic landing of drones using artificial intelligence image recognition," *Int. J. Mech. Eng. Robot. Res*, vol. 11, pp. 357–364, Jun. 2022.

[50] Y. Son, H. Shin, D. Kim, Y. Park, J. Noh, K. Choi, J. Choi, and Y. Kim, "Rocking drones with intentional sound noise on gyroscopic sensors," in *Proceedings of the 24th USENIX Security Symposium (Security)*, Washington, DC, Aug. 2015.

[51] H. Kim, R. Bandyopadhyay, M. O. Ozmen, Z. B. Celik, A. Bianchi, Y. Kim, and D. Xu, "A systematic study of physical sensor attack hardness," in *Proceedings of the 45th IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2024.

[52] J.-H. Jang, M. Cho, J. Kim, D. Kim, and Y. Kim, "Paralyzing drones via emi signal injection on sensory communication channels.," in *Proceedings of the 2023 Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2023.

[53] S. Sönmez, M. J. Rutherford, and K. P. Valavanis, "A survey of offline-and online-learning-based algorithms for multirotor uavs," *Drones*, vol. 8, no. 4, p. 116, Mar. 2024.

[54] *Skydio Autonomy*. [Online]. Available: https://www.skydio.com/skydio-autonomy/.

[55] *Wing Drone Delivery*. [Online]. Available: https://wing.com/.

[56] E. Golinko and X. Zhu, "Generalized feature embedding for supervised, unsupervised, and online learning tasks," *Information Systems Frontiers*, vol. 21, pp. 125–142, Apr. 2019.

[57] G. Simantiris, K. Bacharidis, and C. Panagiotakis, "Closing the domain gap: Can pseudo-labels from synthetic uav data enable real-world flood segmentation?" *Sensors*, vol. 25, no. 12, p. 3586, Apr. 2025.

[58] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning.," *arXiv preprint arXiv:abs/1712.05526*, 2017. [Online]. Available: https://arxiv.org/abs/1712.05526.

[59] Y. Wang and K. Chaudhuri, "Data poisoning attacks against online learning," *arXiv preprint arXiv:1808.08994*, 2018. [Online]. Available: https://arxiv.org/abs/1808.08994.

[60] *AutoPilot*. [Online]. Available: https://github.com/yash1017/Auto_Pilot.

[61] *PX4-ROS2-Gazebo-YOLOv8n*. [Online]. Available: https://github.com/monemati/PX4-ROS2-Gazebo-YOLOv8.

[62] *NVIDIA CUDA Toolkit*. [Online]. Available: https://developer.nvidia.com/cuda/toolkit.

[63] D. Song, K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, and T. Kohno, "Physical adversarial examples for object detectors," in *12th USENIX workshop on offensive technologies (WOOT 18)*, Baltimore, MD, Mar. 2018.

[64] Y. Zhang, Y. Zhang, J. Qi, K. Bin, H. Wen, X. Tong, and P. Zhong, "Adversarial patch attack on multi-scale object detection for uav remote sensing images," *Remote Sensing*, vol. 14, no. 21, Sep. 2022.

[65] J. Noh, Y. Kwon, Y. Son, H. Shin, D. Kim, J. Choi, and Y. Kim, "Tractor beam: Safe-hijacking of consumer drones with adaptive gps spoofing," *ACM Transactions on Privacy and Security (TOPS)*, vol. 22, no. 2, pp. 1–26, May 2019.

[66] R. Akella, H. Tang, and B. M. McMillin, "Analysis of information flow security in cyber–physical systems," *International Journal of Critical Infrastructure Protection*, vol. 3, no. 3-4, pp. 157–173, Jun. 2010.

[67] D. Wanner, H. A. Hashim, S. Srivastava, and A. Steinhauer, "Uav avionics safety, certification, accidents, redundancy, integrity, and reliability: A comprehensive review and future trends," *Drone Systems and Applications*, vol. 12, pp. 1–23, May 2024.

[68] D. Wang, S. Li, G. Xiao, Y. Liu, Y. Sui, P. He, and M. R. Lyu, "An exploratory investigation of log anomalies in unmanned aerial vehicles," in *Proceedings of the 46th International Conference on Software Engineering (ICSE)*, Ottawa, Canada, Apr. 2024.

[69] M. Ouadah and F. Merazka, "Securing uav communication: Authentication and integrity," in *2024 11th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, Leeds, United Kingdom, Jul. 2024.

[70] *DroneXtract*. [Online]. Available: https://github.com/ANG13T/DroneXtract.

[71] Y. Kim, K. Cho, and S. Kim, "Challenges in drone firmware analyses of drone firmware and its solutions," *IEEE Access*, Jul. 2024.

[72] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International conference on machine learning*, Sydney, Australia, Aug. 2021.

[73] *Captum · Model Interpretability for PyTorch*. [Online]. Available: https://captum.ai/.

[74] S. Dinesh, N. Burow, D. Xu, and M. Payer, "Retrowrite: Statically instrumenting cots binaries for fuzzing and sanitization," in *Proceedings of the 41st IEEE Symposium on Security and Privacy (S&P)*, Virtual Conference, May 2020.

[75] E. Bauman, Z. Lin, K. W. Hamlen, *et al.*, "Superset disassembly: Statically rewriting x86 binaries without heuristics.," in *Proceedings of the 2018 Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2018.

[76] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, Zurich, Switzerland, Sep. 2014.

[77] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How trlevkovits2019realansferable are features in deep neural networks?" In *Advances in neural information processing systems*, Montréal, Canada, Dec. 2014.

[78] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "Abs: Scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the 26th ACM Conference on Computer and Communications Security (CCS)*, London, UK, Nov. 2019.

[79] *IDA Free - Hex-Rays*. [Online]. Available: https://hex-rays.com/ida-pro.

[80] *PyTorch*. [Online]. Available: https://pytorch.org/.

[81] *Iris Quadrotor*. [Online]. Available: https://github.com/PX4/PX4-SITL_gazebo-classic/tree/main/models/iris.

[82] *Ardupilot Companion Computers*. [Online]. Available: https://ardupilot.org/dev/docs/companion-computers.html.

[83] *PX4 Companion Computers*. [Online]. Available: https://docs.px4.io/main/en/companion_computer/.

[84] T. Q. Khoi, N. A. Quang, and N. K. Hieu, "Object detection for drones on raspberry pi potentials and challenges," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 1109, Jan. 2021, p. 012 033.

[85] M. T. Topalli, M. Yilmaz, and M. F. Corapsiz, "Real time implementation of drone detection using tensorflow and mobilenetv2-ssd," in *2021 7th International Conference on Electrical, Electronics and Information Engineering (ICEEIE)*, Malang, Indonesia, Oct. 2021.

[86] X. Deng, B. Khan, C. P. Lim, and M.-Y. Liao, "A mobilenetv2-cbam-based model for forest fire classification using uav imagery," in *2023 IEEE 4th International Conference on Pattern Recognition and Machine Learning (PRML)*, Urumqi, China, Aug. 2023.

[87] W. Li, H. Chen, and Z. Shi, "Semantic segmentation of remote sensing images with self-supervised multitask representation learning," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 6438–6450, Jun. 2021.

[88] S. Samaras, E. Diamantidou, D. Ataloglou, N. Sakellariou, A. Vafeiadis, V. Magoulianitis, A. Lalas, A. Dimou, D. Zarpalas, K. Votis, *et al.*, "Deep learning on multi sensor data for counter uav applications—a systematic review," *Sensors*, vol. 19, no. 22, p. 4837, Sep. 2019.

[89] D. Wang, W. Li, X. Liu, N. Li, and C. Zhang, "Uav environmental perception and autonomous obstacle avoidance: A deep learning and depth camera combined solution," *Computers and Electronics in Agriculture*, vol. 175, p. 105 523, Aug. 2020.

[90] D. S. Levkovits-Scherer, I. Cruz-Vega, and J. Martinez-Carranza, "Real-time monocular vision-based uav obstacle detection and collision avoidance in gps-denied outdoor environments using cnn mobilenet-ssd," in *Mexican International Conference on Artificial Intelligence*, Xalapa, Mexico, Oct. 2019.

[91] J. Wang, H. Dai, B. Zhang, S. Qin, J. Zhao, and Z. Zhang, "Plrut: Pseudo label and re-detection boosted unsupervised tracking of unmanned aerial vehicle objects," in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, Urumqi, China, Nov. 2024.

[92] S. Hu, C.-H. Liu, J. Dutta, M.-C. Chang, S. Lyu, and N. Ramakrishnan, "Pseudoprop: Robust pseudo-label generation for semi-supervised object detection in autonomous driving systems," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, New Orleans, LA, USA, Jun. 2022.

[93] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249–289, Oct. 2021.

[94] T. Gu, B. DolanGavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain.," *arXiv prepirint arXiv:abs/1708.06733*, 2017. [Online]. Available: http://arxiv.org/abs/1708.06733.

[95] *Both black boxesfound from plane that crashed in deep gorge in Nepal, killing scores*. [Online]. Available: https://www.cbsnews.com/news/nepal-plane-crash-both-black-boxes-found-deep-gorge-scores-dead/.

[96] *Plane crashes: Last words of pilots captured on black box recordings in air disasters*. [Online]. Available: https://www.nzherald.co.nz/world/plane-crashes-last-words-of-pilots-captured-on-black-box-recordings-in-air-disasters/JNR3UQCHDYT7WFP35W6ODL6LXA/.

[97] *Loganalyzer: Diagnosing problems using logs for ardupilot*. [Online]. Available: https://ardupilot.org/copter/docs/common-diagnosing-problems-using-logs.html.

[98] D. Clark, C. Meffert, I. M. Baggili, and F. Breitinger, "Drop (drone open source parser) your drone: Forensic analysis of the dji phantom iii.," *Digital Investigation*, vol. 22 Supplement, S3–S14, Aug. 2017.

[99] U. Jain, M. Rogers, and E. T. Matson, "Drone forensic framework: Sensor and data identification and verification," in *Proceedings of the 2017 IEEE Sensors Applications Symposium (SAS)*, Glassboro, NJ, USA, Mar. 2017.

[100] B. Kasikci, B. Schubert, C. Pereira, G. Pokam, and G. Candea, "Failure sketching: A technique for automated root cause diagnosis of in-production failures," in *Proceedings of the 25th ACM Symposium on Operating Systems Principles (SOSP)*, Monterey, CA, Oct. 2015.

[101] J. Xu, D. Mu, X. Xing, P. Liu, P. Chen, and B. Mao, "Postmortem program analysis with hardware-enhanced post-crash artifacts," in *Proceedings of the 26th USENIX Security Symposium (Security)*, Vancouver, BC, Canada, Aug. 2017.

[102] S. Park, Y. Zhou, W. Xiong, Z. Yin, R. Kaushik, K. H. Lee, and S. Lu, "Pres: Probabilistic replay with execution sketching on multiprocessors," in *Proceedings of the 22nd ACM Symposium on Operating Systems Principles (SOSP)*, Big Sky, MT, Oct. 2009.

[103] S. Sierla, B. M. O'Halloran, T. Karhela, N. Papakonstantinou, and I. Y. Tumer, "Common cause failure analysis of cyber–physical systems situated in constructed environments," *Research in Engineering Design*, vol. 24, pp. 375–394, Jun. 2013.

[104] M. Shafiee, Z. Zhou, L. Mei, F. Dinmohammadi, J. Karama, and D. Flynn, "Unmanned aerial drones for inspection of offshore wind turbines: A mission-critical failure analysis.," vol. 10, no. 1, p. 26, Feb. 2021.

[105] C. Rudin and J. Radin, "Why are we using black box models in AI when we don't need to? A lesson from an explainable ai competition," *Harvard Data Science Review*, vol. 2, no. 1, Nov. 2019.

[106] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models.," *ACM Computing Surveys*, vol. 51, no. 5, 93:1–93:42, Sep. 2019.

[107] C. B. Azodi, J. Tang, and S.-H. Shiu, "Opening the black box: Interpretable machine learning for geneticists," *Trends in Genetics*, vol. 36, no. 6, pp. 442–455, Jun. 2020.

[108] H. Kim, M. O. Ozmen, A. Bianchi, Z. B. Celik, and D. Xu, "PGFUZZ: Policy-guided fuzzing for robotic vehicles.," in *Proceedings of the 2021 Annual Network and Distributed System Security Symposium (NDSS)*, Virtual Conference, Feb. 2021.

[109] H. Kim, M. O. Ozmen, Z. B. Celik, A. Bianchi, and D. Xu, "Pgpatch: Policy-guided logic bug patching for robotic vehicles," in *Proceedings of the 43rd IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2022.

[110] D. Davidson, H. Wu, R. Jellinek, V. Singh, and T. Ristenpart, "Controlling UAVs with sensor input spoofing attacks," in *Proceedings of the 10th USENIX Workshop on Offensive Technologies (WOOT)*, Austin, TX, Aug. 2016.

[111] S. Kim, M. Liu, J. J. Rhee, Y. Jeon, Y. Kwon, and C. H. Kim, "Drivefuzz: Discovering autonomous driving bugs through driving quality-guided fuzzing," in *Proceedings of the 29th ACM Conference on Computer and Communications Security (CCS)*, Los Angeles, CA, Nov. 2022.

[112] E. Mantas and C. Patsakis, "Gryphon: Drone forensics in dataflash and telemetry logs," in *International Workshop on Security*, Tokyo, Japan, Aug. 2019.

[113] U. Challita, A. Ferdowsi, M. Chen, and W. Saad, "Machine learning for wireless connectivity and security of cellular-connected uavs," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 28–35, Feb. 2019.

[114] M. Y. Alzahrani, "Enhancing drone security through multi-sensor anomaly detection and machine learning," *SN Computer Science*, vol. 5, no. 5, p. 651, Jun. 2024.

[115] *Zenodo*. [Online]. Available: https://zenodo.org/records/17945698.

A ML Model Analysis Limitations

FIRA's forensic conclusions depend on the accuracy of the underlying neuron attribution techniques, and misidentification of influential neuron groups could lead to inaccurate root cause determination. Additionally, FIRA performs key contributing neuron groups during the pre-flight phase based on the ground station model and test dataset, which may miss neuron groups that become influential during online learning. The neuron groups strategy also varies across network architectures, and different model structures may require different grouping methods for effective monitoring. The current implementation focuses exclusively on convolutional neural networks. For other CNN-based models such as YOLO, our approach directly applies since they share similar convolutional architectures for feature extraction. For transformers, a similar approach could group by attention heads and apply attribution methods to identify key contributing heads. Validation of these extensions remains as future work.

B Evaluation Generalization

FIRA has been implemented and evaluated only on the OpenUAV, PX4, ArduPilot, Gazebo and ROS2. Generalization to other platforms would require adapting the inter-module causal graph generation to different middleware architectures. All experiments were conducted in simulation rather than on physical hardware. The evaluation considers only vision-based ML models; other sensor such as LiDAR-based perception or IMU-based learning systems were not evaluated.

C Applying FIRA in Real-World Missions

For each mission, training data would first be collected by manually flying the physical UAV without the online learning model attached, collecting camera frames from the onboard camera. The ground station model would be trained with the collected training data following the same procedure as in §4. The UAV would then fly the missions autonomously with the ground station model, collecting and labeling new data during flight using the same online learning implementations as in §4.1. To execute attacks, adversaries would introduce visual triggers in the physical operational environment during online learning, following the same attack methodology as in §4.1, based on prior research [94]. The ground station would receive neuron group updates and module data over actual RF telemetry links within the bandwidth constraints. After induced crashes, FIRA would perform forensic analysis using the data logged at the ground station to identify root causes.