Stateful Triage for Reliable and Secure Wildfire Monitoring at the Edge

Long Tran*, River Bartz†, Rajesh Sankaran‡, Ulrich Kremer*, Ramakrishnan Durairajan†, Sudarsun Kannan*
*Rutgers University, USA,†University of Oregon, USA, ‡Argonne National Laboratory, USA

***Abstract*—Wildfire monitoring is increasingly done at the network edge using small, shared (multi-tenant) sensor nodes with cameras and on-device ML. While this architecture scales, it is also susceptible to new reliability and security risks. In this paper we study how co-located workloads and side channels can undermine wildfire detection on a shared GPU edge node.**

Using real workloads, we show two key effects. First, inference latency is *content-dependent*: frames that look like fire or smoke tend to trigger heavier model paths and take longer to process. Second, a concurrent ML application (e.g., an animal detector) can slow the wildfire pipeline and cause instability by competing for VRAM and compute. Taken these effects together, timing and simple GPU telemetry (utilization, memory, power) leak enough signal for a co-tenant to *infer* when "fire-like" frames are being processed and to time bursts of activity that further delay or drop those frames.

To mitigate this problem, we argue that edge deployments should treat *stateful triage* as a first-class system component. Rather than emitting isolated detections, a lightweight triage layer maintains short event state, enforces basic spatiotemporal consistency (e.g., smoke before flame, persistent growth), and adapts resource use based on both event state and platform health. This paper contributes: (1) an empirical analysis of multi-tenant risks for wildfire ML at the edge, including content-dependent latency and timing side channels; (2) a security-oriented threat model for shared edge nodes; and (3) a design sketch for stateful triage that can reduce false positives, prioritize likely true events, and blunt denial-of-service and timing leaks.

I. Introduction

Wildfires are increasing in frequency, intensity, and unpredictability, threatening ecosystems, infrastructure, and human life. Early detection and rapid response are therefore essential.

Modern wildfire monitoring relies on cross-boundary, distributed architectures that span edge devices, wide-area networks (WANs), and cloud backends. In a typical deployment, edge nodes (e.g., environmental monitoring stations equipped with multispectral cameras, thermal imagers, microphones, and meteorological sensors) continuously observe local conditions. These platforms often execute multiple sensing pipelines/applications concurrently (e.g., wildfire detection, air-quality monitoring, bioacoustics, wildlife detection) under tight power, compute, and storage budgets. Captured telemetry traverses WAN backbones to cloud services, where machine learning (ML) models perform event classification and human operators review alerts. This human-in-the-loop design introduces unavoidable latency (e.g., ranging from minutes to hours) depending on network conditions and operator availability.

Because these nodes are shared across different sensing projects, governmental agencies, and research teams, they are effectively *multi-tenant*: heterogeneous pipelines/applications and owners co-exist on the same constrained edge node and contend for CPU, memory, storage, radios, and accelerators. This opens up two security vulnerabilities.

First, a common yet fragile assumption in such deployments is that per-frame inference cost is roughly uniform. In reality, as we show in § III-A, frames containing wildfire cues (e.g., smoke plumes, flame-like textures, haze) systematically exercise heavier resource usage (e.g., GPUs and WAN) and post-processing paths than benign frames. The result is a repeatable latency gap between hazard-positive vs. benign content that aligns with compute and thermal utilization peaks. In a multi-tenant setting, this content–latency correlation is not merely performance variability; it becomes an adversarially exploitable signal.

Concretely, an untrusted co-tenant (e.g., adversary) can infer when the wildfire detector is processing likely fire-positive frames. They might do this by probing kernel queueing or sampling coarse GPU telemetry. Once they know the detector is engaged in these critical computations, the adversary could deliberately launch short, intensive workloads on the shared hardware (e.g., GPUs). These bursts of activity would compete for resources, thereby slowing down the detector's processing or even blocking it altogether, exactly at those moments when rapid alerts are most important. Beyond GPU and CPU activity, a sophisticated adversary might also use other side channels such as power and thermal fingerprints, as well as activities in shared caches, buses, and I/O channels, to infer and interfere with the detector's processing of wildfire-like content. Overall, these *side channels* are difficult to eliminate under tight resource and energy budgets.

Second, identifying wildfire events (through side channels) can also trigger denial-of-service (DoS) vectors such as compute/accelerator exhaustion, memory-pressure causing out-of-memory (OOM) events, battery and thermal throttling, and storage failures. We show through experiments (in § III-A) that side-channel exposure elevates both the availability and confidentiality risks of edge wildfire monitoring.

More specifically, a misconfigured or malicious co-tenant can monopolize CPU and GPU cycles by using oversized batches, parallel inference, or bursty inference, starving wildfire detection. We refer to this as *compute/accelerator exhaustion*. Similarly, an attacker can induce memory pressure on

both GPU and CPU through tensor bloat, leading to memory thrashing or OOM kills and further inflate latency. On solar- or battery-powered edge devices, sustained processing and network storms can drain batteries, causing brownouts and blind spots (*energy starvation*). Frequent writes and insufficient cooling push SSDs and GPUs into throttling or failure; an adversary can amplify wear and heat with bursty capture and upload patterns (referred to as *storage/thermal DoS*).

Position: We argue that wildfire monitoring at the edge should treat *stateful triage* as a first-class system component to mitigate these security vulnerabilities. Given the limited computational resources and constrained energy budget of edge platforms, it is critical to prevent complex side-channel attacks from escalating into denial-of-service incidents that disrupt wildfire detection, particularly during time-critical events.

Toward this goal, rather than performing independent, stateless per-frame wildfire detections from the edge, we propose a lightweight triage layer that (i) tracks event hypotheses over short temporal windows, (ii) enforces wildfire-consistent dynamics (e.g., smoke preceding flame, spatial growth rather than abrupt disappearance, cross-sensor agreement, and (iii) actively manages resources in response to event state and platform health. Our key hypothesis is that by coupling spatiotemporal reasoning with resource awareness, the stateful triage can suppress isolated false positives, prioritize likely true events, and minimize the operational impact of the co-tenant interference and side-channel timing effects.

This paper makes three contributions:

- *Threat identification.* We identify a new class of side-channel and denial-of-service (DoS) attacks, with particular emphasis on multi-tenant edge deployments, where time-critical applications like wildfire monitoring are co-run with other workflows that share processing infrastructure like compute, memory, storage, and network.
- *Threat demonstration.* We use real-world deployed applications and datasets to demonstrate threats through experiments and measurements. Specifically, using YOLO, a computer vision framework for wildfire detection and wildlife monitoring systems [1], [2], we demonstrate that content-dependent latency is systematic and that co-residency induces consistent slowdowns and instability.
- *Stateful triage.* To mitigate these threats, we propose a lightweight triage layer that maintains state, fuses multi-sensor signals, and applies temporal consistency checks. By encoding spatiotemporal progression rules, this system suppresses isolated false positives, assigns confidence scores, and directs operator attention to likely true wildfire events. We also outline the challenges of realizing a full-fledged triage layer but leave the implementation details for future work.

II. Background and Related Work

Research on wildfire monitoring spans sensing hardware, computer vision pipelines, and distributed system design at the edge–cloud continuum. Security work on IoT/edge deployments is rich, but most efforts treat security separately from end-to-end monitoring workflows and rarely evaluate under the multi-tenant, resource-constrained realities of deployed systems. We group related work into five areas and then discuss state-of-the-art (SOTA) hazard monitoring systems, highlighting what remains missing.

Wildfire sensing systems and datasets. Operational camera networks and environmental observatories deploy RGB/IR cameras and meteorological sensors to detect smoke or flame and dispatch alerts to human operators (e.g., regional fire-camera grids and public safety observatories) [3]. Open datasets and baselines for vision models, including RGB and thermal imagery, have accelerated progress; notable examples include OpenFire and Pyro-SDIS, which provide labeled images for smoke/flame detection and are common benchmarks for CNN/YOLO variants [4]. UAV-based and satellite/EO approaches (e.g., MODIS, VIIRS, Sentinel-2) complement fixed sensors with wide-area coverage and periodic revisit, and there is growing interest in TinyML [5], on-device inference for remote deployments [5], [6].

Limitations. Most of these efforts emphasize detection accuracy rather than security, resilience and availability. They seldom model adversaries targeting resource exhaustion, and they rarely report robustness under multi-tenant contention, battery pressure, or storage throttling. Dataset papers typically benchmark precision/recall in isolation rather than end-to-end availability in the face of attacks or faults.

Edge/IoT security and DoS on constrained nodes. AroMa [7] studies edge accelerators and DoS via bit flips and RowHammer-induced memory corruption, focusing on DNN vulnerabilities. EdgeOS [8] targets multi-tenant edge systems, aiming to prevent tenant-to-tenant attacks through network isolation. Its strict requirements that demands not using file systems and requiring a new OS process abstraction, which makes it hard to deploy and support diverse hazard applications. The challenges in wildfire monitoring involving GPUs, CPUs, memory, and storage differ from those considered by EdgeOS. Classic IoT security work has examined denial-of-sleep (e.g., forcing radios to remain active), computational DoS (e.g., CPU spinning), GPU compute, and memory exhaustion, and routing or protocol misuse [9]

Limitations. There has been little focus on *co-located* ML pipelines on shared GPUs and their unique resource coupling (e.g., driver queues, unified memory pressure). Likewise, energy-centric defenses (e.g., duty cycling, rate limiting) in multi-tenant edge systems have not addressed the end-to-end challenges that span software and hardware. Real-world defenses like containerization [3] for resource and rate-limiting solutions are prone to well-documented isolation and security issues, even in datacenter settings.

Storage, thermal, and reliability. To the best of our knowledge, little prior work has focused on storage reliability or thermal security issues in edge-based systems. We believe our study is the first to examine storage reliability and

security in the context of hazard monitoring. Our analysis shows that, to support multi-tenant ML pipelines, workloads are generally write-intensive: continuous frames, thumbnails, feature caches, and logs generate sustained writes on SD and NVMe media.

Limitations. Prior data-centric research has documented issues such as SSD wear, thermal throttling, and firmware pathologies under mixed workloads. Yet, both the attack models and mitigation strategies from datacenters are hard to adapt to wildfire monitoring, where tasks cannot be paused or offloaded. Further, temperatures can rise sharply or fluctuate unpredictably during active fire or smoke events, precisely when workflows must remain highly resilient. To our knowledge, no studies have explored integrating SMART or thermal telemetry for storage to proactively adapt workload behavior.

Side-Channel leakage on shared hardware. A huge body of work has examined side-channel attacks on multi-tenant datacenter and edge AI environments. In the context of edge AI, where hardware resources are limited, sharing is unavoidable. Applications contend for time-shared CPUs and GPUs, and space-shared memory, storage, caches, TLBs, and communication paths such as PCIe queues, channels, and power sources. Sharing results in identifying/fingerprinting application activity. In the context of hazard monitoring systems that deploy applications in containers that share OS resources and hardware, they are prone to direct and indirect resource monitoring and mounting side-channel attacks.

Limitations. While prior studies provide useful, broadly applicable insights, we show that side-channel vulnerabilities (e.g., GPU resource utilization) can be leveraged to launch denial-of-service (DoS) attacks on system resources. In wildfire monitoring systems, however, defenses must be context-specific and stateful, able to operate under strict power constraints without sacrificing detection accuracy, recall, or timeliness. Moreover, prior work has not evaluated whether wildfire tasks reveal distinctive "signatures" (e.g., longer per-frame inference for smoke-like frames) or how such signatures can be concealed within stringent energy budgets in multi-tenant settings. Closing this gap is a central focus of our study.

Human-in-the-loop triage and adversarial ML. Hazard monitoring often combines one-shot image classification or detection with human vetting. A recent position paper [10] shows that explanation maps are vulnerable to adversarial manipulations, undermining model trust across domains such as security, medical imaging, and critical infrastructure. It advocates using human visual attention models to improve resilience. While orthogonal and valuable, these approaches do not address the core challenge of preemptive, automatic threat detection. Cognitive overload remains a concern during active events or when continuous human resources are unavailable.

Limitations. Most alerting pipelines are stateless and ignore spatio-temporal consistency (e.g., smoke preceding flame, spatial growth, cross-sensor agreement). Few propose lightweight, on-edge triage that fuses sensors to suppress isolated false positives while maintaining accuracy and low latency.

A. State-of-the-Art Hazard Monitoring Systems

Beyond wildfire, SOTA hazard monitoring systems span earthquakes, landslides, floods, volcanoes, and air quality. Large-scale programs such as ShakeAlert for earthquakes, WIFIRE for fire analytics, and multi-hazard observatories (e.g., OOI/NOAA assets) illustrate end-to-end telemetry, modeling, and alerting [11], [12]. Recent proposals emphasize:

- **Spatiotemporal deep learning.** 3D CNNs, ConvLSTMs, and transformers (ViT/Swin/TimeSformer) for long-horizon event inference and nowcasting on multimodal streams (video, thermal, meteorology) [13].
- **Edge-first analytics.** On-device compression paired with cloud processing used for filtering of videos and events captured on cameras through fast ML (e.g., TinyML) to reduce network bandwidth use [5].
- **Learning at the edge.** Federated and continual learning to adapt to local conditions while preserving privacy; active learning loops with operator feedback [14].
- **Resource/security awareness.** Workload isolation and resource partitioning and rate limiting through virtualization, containerization, enclaves/TEEs for model and data protection; GPU partitioning when possible [15].

Limitations. We observe that state-of-the-art systems generally focus on evaluating independent hardware or software components rather than end-to-end hazard monitoring systems such as wildfire monitoring, especially under multi-tenant contention. Given the strict energy constraints, it is critical to quantify the trade-offs of different security prevention techniques within these limited budgets.

B. Fundamental Limitations and Security Requirements

Several aforementioned limitations motivate the following requirements (**R**s). Three, in particular, stand out:

(R1) Security integrated with resource-aware operation. We lack designs that co-optimize security with compute/memory/energy/storage budgets typical of remote, battery-backed nodes. Defenses like strict resource isolation are often too expensive; systematic evaluations quantifying their real cost/benefit for wildfire tasks.

(R2) Multi-tenant isolation and side-channel challenges for edge AI. There is no widely adopted approach for per-resource limits (CPU shares, GPU partitioning when available, I/O and network QoS) on commodity edge boxes, nor for side-channel-aware scheduling for camera/thermal workloads.

(R3) Stateful, physics-aware triage in the loop. Lightweight models that enforce wildfire detection besides running analysis from multiple sensors are not widely studied. This could overload and break several DoS prevention strategies (e.g., preventing spurious NVMe/SSD writes).

Our work addresses these requirements by (i) characterizing DoS/DDoS vectors against compute, memory, energy, storage, and networks in multi-tenant wildfire nodes; (ii) proposing a

side-channel and DOS-aware isolation and scheduling based on realistic hardware; and (iii) introducing a preliminary design of a stateful triage layer to suppress false positives, reduce write amplification and energy drain, and reduce the stress on the human-in-the-loop.

III. Motivation

A. Content-Dependent Latency Vulnerability

Wildfire detection models deployed at the edge are often assumed to have a uniform per-frame inference cost. Our measurements show that this assumption does not hold: frames containing wildfire cues (e.g., smoke plumes, flame-like textures) consistently induce more non-maximum suppression (NMS) costs, reported performance issues in models before YOLOv10 [16]. These effects result in a measurable, repeatable latency gap between *fire* and *no-fire* frames.

Figure 1 quantifies this effect for a YOLO detector on a single GPU. We use YOLO-based detection as the state-of-the-art vision + AI-based wildfire detection approach used across multiple deployed systems, such as SAGE and OHAZ. We plot per–image latency and highlight the globally slowest and fastest K frames. In this representative run, *all* slowest K frames are fire/smoke positives are clustered around $\approx 55ms$, while the fastest K frames are between $12-15ms$ range. The medians (dashed lines) separate clearly: positives at ~ 51 ms versus negatives at ~ 35 ms, a $\approx 1.45\times$ median slow–down. Thus, the positive and negatives differ by roughly $3-4\times$, with those with higher fire/smoke frames always at the higher end. We observe the same pattern across different parts of the workload; fast-fire frames (a red "x") appear when detections are small and the NMS work on these frames is light.

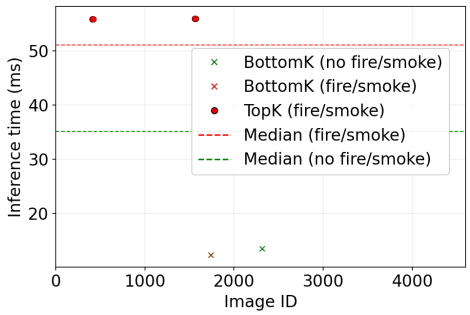

Fig. 1: Per-image latency. Red: Top-K slowest (fire). Green: Bottom-K fastest (no-fire).

This result highlights a critical attack vector. In what follows, we describe the attack vector in detail, along with a threat model and a step-by-step attack procedure. We leave the demonstration and empirical validation to future work.

Attack vector: Content inference via side channels. In a multi-tenant edge environment, a co-resident process can exploit this correlation without reading camera frames or model outputs. An attacker samples timing and hardware counters (e.g., NVML GPU utilization, memory usage, power draw) or measures queueing delays using their kernels (e.g., NVIDIA Nsight Systems tools) By using these observations with the node's fixed frame-capture rate and training a simple classifier, the attacker can infer when the wildfire detector is processing a fire-positive frame. This transforms resource (GPU) telemetry into a *content inference channel* that compromises the confidentiality of hazard detections.

Threat model: Timing and GPU-utilization side channels. We consider an adversary that is a co-tenant on the same edge node, such as another analytics workload or an untrusted container. These adversaries' programs/applications cannot directly access raw images or detector outputs from the victim application. Instead, the adversary relies on indirect measurements obtained from their own workloads and from system-level telemetry. For example, an attacker could launch short GPU kernels or real application workloads (e.g., wildlife detection during wildfire detection events) that are queued alongside the victim's inference jobs, and the adversary can measure the latency of their own kernel executions. They can also monitor GPU utilization, memory usage, and power usage using tools such as `nvidia-smi` or the NVIDIA Management Library (NVML). In addition, the adversary could observe PCIe queueing delays (for storage) and track the end-to-end API response times of the victim's service, that is, detect if a video or a set of frames has images with wildfire or not. These inferences, when combined, can provide enough information without requiring direct access to sensitive data such as wildfire images or video frames.

Attack: Fingerprinting procedure. An attacker continuously launches GPU kernels with execution times of approximately 1–2 ms and records their service times. In parallel, the adversary samples GPU telemetry via NVML at a high frequency, typically between 10 Hz and 50 Hz, to capture fine-grained fluctuations in utilization and memory usage. One could align the measurement with the camera frame rate used by the victim application. An attacker can associate resource usage patterns with specific inference intervals, such as "fire" events, and a labeled dataset is used to train a lightweight classifier. As seen in Figure 1, the results show that Top-K frames (that are generally fire events), corresponding to high-latency intervals, can be reliably distinguished from Bottom-K frames using GPU utilization features. This enables an adversary to perform fire/no-fire inference without ever accessing the images.

B. Denial-of-Service Attack

The "fire-now" detection windows resulting from the side channel described above enable a new form of targeted denial-of-service (DoS) attack. By injecting GPU- or memory-intensive bursts during these critical periods, an adversary can inflate median or tail latency by more than 50%, increase frame drops and missed detections, and even trigger out-of-memory (OOM) crashes due to VRAM exhaustion or fragmentation. Such attacks can significantly degrade wildfire monitoring and detection during actual wildfire events.

Attack vector: Targeted denial-of-service. An adversary exploits knowledge of when fire-positive frames are being processed (via timing side channels in § III-A) and injects disruptive workloads precisely during these windows. Unlike

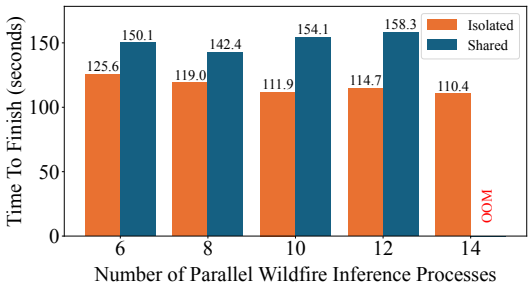

Fig. 2: Wildfire detection (victim) runtime when varying the number of parallel processes. Blue: isolated. Orange: co-located with the wildlife monitoring pipeline. Co-located run at $N=14$ shows CUDA OOM.

a continuous flooding attack, such targeted DoS focuses resources at the most critical intervals, amplifying its impact on latency and reliability. More critically, a well-timed DoS can delay detections, induce frame loss, or force application crashes, thereby undermining the operational availability of the wildfire detection system.

Threat model: Co-resident adversary. We assume an adversary shares the same edge GPU node as another legitimate workload within an untrusted container. The adversary cannot access images or model outputs directly, but runs its own GPU-intensive processes (e.g., image classification) that contend with the victim's fire inference. The adversary continuously probes GPU availability with lightweight kernels or monitors telemetry (e.g., utilization, memory, queueing delays). When fire-positive intervals are detected, the adversary launches GPU- or memory-heavy tasks (e.g., large batch inference) that overlap with the victim's inference window. This increases the victim's per-frame latency by more than 50%, triggers frame drops, and pushes the victim into CUDA OOM errors. Such attacks are harder to detect and mitigate as they don't fully monopolize the GPUs at all times.

Demonstration of the attack. We demonstrate this DoS attack by measuring the impact of a co-located ML workload on wildfire detection in a shared edge GPU setting, attributing slowdowns to VRAM pressure and timing side channels.

In our demonstration, we launch up to $N \in \{2, \ldots, 24\}$ YOLO wildfire-detection processes (same model and code) that share the image shards (split equally across processes) on a single GPU. When run in isolation, the workload scales smoothly until saturation of the GPU resources, primarily due to GPU memory usage. In our GPUs, a batch size of 15 images for inference provides the fastest inference of $110.4ms$ from 126 at $N=6$. Beyond these values, the runtime significantly starts to increase, followed by application failures.

Adversary. A continuously running wildlife monitoring pipeline (Microsoft MegaDetector v6, YOLOv10-Compact) processes a 22,566 image dataset in large batches (100 images/step), maintaining 2.8 GiB VRAM residency and steady GPU activity. This simulates realistic multi-application edge deployments. We compare isolation versus co-location across $N \in \{6, 8, 10, 12, 14\}$ at batch size 15. Each trial is repeated 5 times. Metrics include per-image latency (pre/infer/post) and throughput. Frames are labeled fire/no-fire for latency analysis.

Results. Figure 2 shows that co-location inflates wildfire inference time by 20% at low concurrency ($N = 6, 8$) and up to 38%. At $N = 15$, the victim runs cleanly in isolation but fails with CUDA OOM under co-location. This shows that a simple adversary based on memory usage is enough to induce both the allocator and the application failure. As discussed earlier, per-frame analysis confirms that the highest latencies occur on fire-positive frames.

IV. Preliminary Design of Stateful Triage Layer

We now describe the preliminary design of a *stateful triage layer* as a first-class system component for reliable and secure wildfire monitoring at the edge, leaving the implementation details for future work. Unlike existing stateless detection pipelines that emit independent per-frame alerts, our triage layer maintains short-term event state, enforces wildfire-consistent dynamics, and adaptively manages system resources in response to event progression and system health.

A. Design Goals

The triage layer is designed to meet three goals. (1) *Reduce false positives.* By incorporating temporal and cross-sensor context, spurious or isolated detections (e.g., transient smoke-like textures, camera noise) are suppressed before alerts are escalated. (2) *Prioritize true events.* Likely wildfire events are promoted based on persistence and spatiotemporal growth, ensuring that scarce system resources and operator attention are allocated to the most critical situations. (3) *Thwart Adversaries.* By coupling detection with resource-awareness, the triage layer minimizes timing and DoS attacks by masking content-dependent latency signatures and combating adversarially induced slowdowns or crashes.

B. Core Components

At the core of triage layer are three components:

(1) Event state tracker. This component tracks hypotheses of wildfire events over a sliding time window (e.g., 5–30 seconds). Rather than treating each frame independently, the tracker aggregates detections, updating an event hypothesis with attributes such as temporal persistence (does the smoke/flame appear consistently?), spatial progression (does the detection region expand or move realistically?), and cross-sensor corroboration (e.g., thermal rise alongside visual smoke). Only events that accumulate sufficient evidence across these dimensions are escalated.

(2) Fire progression rules. Drawing from wildfire physics, the triage layer enforces lightweight progression rules such as "smoke precedes flame," "events grow gradually rather than appearing/disappearing abruptly," and "thermal anomalies must co-occur with visual cues." These rules filter out isolated false detections and impose domain-specific structure on otherwise noisy ML outputs.

(3) Resource-aware adaptation. The triage layer continuously monitors system health (VRAM pressure, GPU/CPU

utilization, thermal state, storage I/O). During high load or suspected adversarial bursts, it adapts resource use by different mechanisms, e.g., downsampling or deferring low-confidence events, throttling non-critical tasks, prioritizing frames associated with ongoing event, among others. This enables maintaining availability and mitigating timing leakage by decoupling latency from alert generation.

C. Operational Flow

The triage workflow proceeds in three stages. (1) *Detection Ingest:* Frame-level predictions from the wildfire detector are passed to the triage layer alongside system telemetry (GPU/CPU/memory usage). (2) *Stateful Fusion:* The triage layer integrates inputs into event hypotheses, enforces progression rules, and updates per-event confidence. (3) *Adaptive Output:* Based on system health, the triage layer either escalates alerts, defers reporting, or suppresses transient positives. Escalated alerts are tagged with confidence metadata, for operator review.

D. Security Benefits

The triage layer strengthens the system against both benign errors and adversarial interference along three dimensions. (1) *Side-channel mitigation:* By smoothing per-frame variability into aggregated state, triage reduces the direct correlation between frame content and instantaneous latency, weakening timing inference channels. (2) *DoS resilience:* By prioritizing persistent, high-confidence events under contention, triage ensures that adversarial bursts disproportionately affect low-value frames rather than critical fire-positive sequences. (3) *Operational robustness:* Triage reduces write amplification to storage and smooths power demand by suppressing spurious detections, extending system longevity in battery-backed or thermally constrained deployments.

E. Deployment Considerations

The triage layer will be lightweight by design. We envision the core logic (event tracking, progression checks, resource monitoring) to require only modest CPU and memory overhead, making it suitable for constrained edge nodes. Interfaces will be modular, i.e., the triage can consume detections from different ML backends (YOLO, transformers, thermal models) and can expose its event hypotheses to both automated alerting systems and human operators. While more advanced versions could incorporate learning-based state machines or cross-node coordination, our initial design will focus on per-node efficiency and resilience.

V. Summary

Wildfire monitoring is increasingly deployed on multi-tenant edge nodes with cameras, sensors, and GPUs. While effective, this setup introduces new security risks. Our paper identifies two key vulnerabilities. (1) Content-dependent latency: wildfire detection models run slower on fire or smoke frames than benign ones, creating a measurable and repeatable latency gap. This variation leaks information and enables side-channel inference of "fire-now" events. (2) Denial-of-service exposure: adversaries sharing the same GPU node can exploit these timing signals to launch bursts of GPU or memory load during critical fire-positive windows. Such attacks inflate latency, drop frames, and may trigger CUDA OOM crashes. To mitigate these risks, the paper proposes a stateful triage layer as a core component of wildfire detection pipelines. Unlike stateless per-frame alerts, the triage layer maintains short-term event hypotheses, enforces wildfire-consistent dynamics (such as smoke preceding flame), and adapts resource usage based on both event state and system health.

References

[1] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[2] A. Hernandez, Z. Miao, L. Vargas, S. Beery, R. Dodhia, P. Arbelaez, and J. M. L. Ferres, "Pytorch-wildlife: A collaborative deep learning framework for conservation," *arXiv preprint arXiv:2405.12930*, 2024.

[3] ANL, "SAGE continuum project," 2024. [Online]. Available: https://sagecontinuum.org/

[4] "Pyro-sdis smoke detection dataset," Hugging Face dataset, 2024. [Online]. Available: https://huggingface.co/datasets/pyronear/pyro-sdis

[5] R. Kallimani, K. Pai, P. Raghuwanshi, S. Iyer, and O. L. A. López, "Tinyml: Tools, applications, challenges, and future research directions," *Multimedia Tools and Applications*, vol. 83, no. 10, p. 29015–29045, Sep. 2023.

[6] W. Schroeder, P. Oliva, L. Giglio, and I. A. Csiszar, "Validation of viirs active fire detection products using landsat-8 and modis data," *Remote Sensing of Environment*, vol. 145, pp. 19–31, 2014.

[7] X. Chen, M. Merugu, J. Zhang, and S. Ray, "AroMa: Evaluating deep learning systems for stealthy integrity attacks on multi-tenant accelerators," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 19, no. 2, pp. 1–17, Apr. 2023.

[8] K. Ren, T. Kim, S. Peter, I. Zhang, and T. Anderson, "Fine-grained isolation for scalable, dynamic, multi-tenant edge clouds," in *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. Boston, MA, USA: USENIX Association, 2020, pp. 927–944.

[9] L. Gu and J. Stankovic, "Radio-triggered wake-up capability for sensor networks," in *Proceedings. RTAS 2004. 10th IEEE Real-Time and Embedded Technology and Applications Symposium, 2004.*, pp. 27–36.

[10] R. McCoppin, M. Kennedy, P. Lukyanenko, and S. Kennedy, "Overcoming adversarial attacks for human-in-the-loop applications," 2023. [Online]. Available: https://arxiv.org/abs/2306.05952

[11] J. J. McGuire, C. W. Ulberg, A. I. Lux, M. Böse, J. R. Andrews, D. E. Smith, B. W. Crowell, J. R. Murray, I. Henson, R. Hartog *et al.*, "Shakealert® version 3: Expected performance in large earthquakes," *Bulletin of the Seismological Society of America*, vol. 115, no. 2, pp. 533–561, 2025.

[12] I. Altintas, J. Block, R. De Callafon, D. Crawl, C. Cowart, A. Gupta, M. Nguyen, H.-W. Braun, J. Schulze, M. Gollner *et al.*, "Towards an integrated cyberinfrastructure for scalable data-driven monitoring, dynamic prediction and resilience of wildfires," *Procedia Computer Science*, vol. 51, pp. 1633–1642, 2015.

[13] V. Sutar, A. Singh, and R. Chandra, "Spatiotemporal deep learning models for detection of rapid intensification in cyclones," 2025. [Online]. Available: https://arxiv.org/abs/2506.08397

[14] H. G. Abreha, M. R. Hayajneh, and M. A. Serhani, "Federated learning in edge computing: A systematic survey," *Sensors (Basel, Switzerland)*, vol. 22, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:245868869

[15] Microsoft Azure, "Confidential computing at the edge," 2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/iot-edge/deploy-confidential-applications

[16] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "Yolov10: Real-time end-to-end object detection," *arXiv preprint arXiv:2405.14458*, 2024.