I-Guard: Real-Time Weapon and Violence Detection

Abstract

Mass shootings and violent incidents have made it imperative to deploy real-time threat detection systems that can identify weapons and aggressive behaviour before harm occurs. I-Guard is an open-source edge AI framework that addresses this need. Running entirely on local hardware, it uses a modular three-stage pipeline to monitor multiple camera streams, detect guns or knives, verify potential threats with a higher-accuracy model and surface alerts to human operators for final decisions. This white paper summarises the motivation behind I-Guard, describes its architecture and features, and provides guidance for configuring and deploying the system.

Introduction

Violence in schools, workplaces and public spaces is an ongoing concern. Traditional surveillance systems capture footage but rely on human operators to notice a threat, often after a critical delay. Automated weapon detection can shorten response times, but many solutions are cloud-based or require expensive hardware. I-Guard is designed to be an accessible starting point for researchers and practitioners who want to build low-latency, on-premises systems. According to the project's README, it runs on Nvidia Jetson devices or other GPU-equipped hosts and implements a **three-stage pipeline** consisting of fast detection, verification and human-in-the-loop escalation 【847524538154419†L0-L18】. The framework aims to reduce false positives while keeping latency low and to give local operators control over escalation decisions.

Design Goals

I-Guard's design is guided by several principles:

- **Low latency and edge deployment.** All inference runs on local hardware; no cloud connectivity is required. Hardware decoding/encoding via GStreamer and NVENC/NVDEC helps keep latency low 【847524538154419†L35-L40】.
- **Modularity.** The system is broken into independent components (camera adapters, detection modules, ring buffers and UI) that can be swapped or extended.
- **Configurability.** Users can adjust detection thresholds, choose different models and set the length of the pre- and post-event buffer via a YAML configuration file 【176380815276105†L19-L37】.
- **Multi-camera support.** A single GPU can handle multiple RTSP or USB streams by time-multiplexing inference 【847524538154419†L29-L31】.
- **Local control.** Events are surfaced to a local web interface where operators can acknowledge, dismiss or trigger escalation 【847524538154419†L16-L20】.

Architecture and Pipeline

Stage 1 – Fast Detection

The first stage of the pipeline uses a lightweight object detector such as YOLOv8 to scan every frame for people, guns and knives【847524538154419†L8-L13】. The FrameDetector class encapsulates this logic and supports two back-ends: Ultralytics' YOLO for development and testing, and TensorRT engines for maximum speed【812088143447372†L9-L18】. The detector runs on each frame and applies heuristic rules to flag candidate events. For example, if any frame contains a 'gun' or 'knife', a weapon flag is set, and a mass_shooter flag is raised when two or more weapons appear【812088143447372†L170-L206】. Placeholder flags exist for pointing, firing, falls and assault, and developers are encouraged to implement their own heuristics【812088143447372†L170-L206】.

Stage 2 – Clip-Based Verification

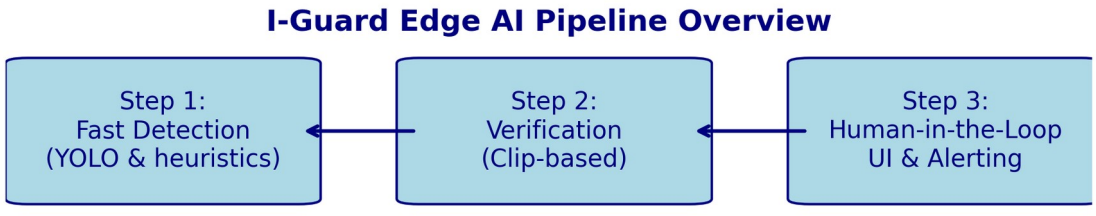
Fast detectors can generate false positives. When Stage 1 flags an event, I-Guard optionally invokes a slower verifier on a short video clip (typically 10 seconds before and after the trigger). The ClipVerifier counts how many frames contain weapon labels and returns a confidence score; if more than half of the frames contain a weapon, the event is considered verified【639396526726164†L47-L76】. In a production system this component could be replaced by a 3D convolutional network or action-recognition model【639396526726164†L47-L76】.

Stage 3 – Human in the Loop

Verified events are not automatically escalated. Instead, they are added to a queue that a local operator can view through a simple web interface. The UI lists recent events, displays the time, camera ID, flags and verification score, and allows operators to play the recorded clip, acknowledge the alert or dismiss it【807561843126669†L16-L44】. Human confirmation helps prevent nuisance alarms and allows context-specific actions.

Pipeline Overview

The figure below summarises the I-Guard pipeline. Video frames from each camera pass through the fast detection stage. When a potential threat is detected, a clip is extracted and sent to the verifier. Events that exceed the verification threshold are surfaced to the user interface for review. The operator can then decide whether to trigger an alarm, contact authorities or ignore the event.



I-Guard pipeline overview

Underlying Components

The pipeline is orchestrated by the InferencePipeline class, which spawns a worker thread for each camera. Each worker reads frames via a CameraAdapter, stores them in a ring buffer and runs the Stage 1 detector【961720882016570†L91-L106】. When a detection is flagged, an Event containing the timestamp, camera ID and flags is enqueued. A separate consumer thread pulls events from the queue, runs Stage 2 verification if enabled, saves the clip to disk and records the event in a history list for display【961720882016570†L163-L223】. The pre- and post-buffer durations and queue size are configurable in config.yaml【176380815276105†L19-L39】.

Key Features

I-Guard offers several features that make it suitable for real-time threat detection on the edge:

Feature	Description	Citation
Multi-camera support	Multiple RTSP/USB streams can be processed concurrently and time-multiplexed on a single GPU【847524538154419†L29-L31】.	README
Plug-and-play models	Bundled with a YOLOv8n model for quick start; users can swap in their own TensorRT engines or ONNX models【847524538154419†L31-L34】.	README
Edge-friendly	Runs entirely on local hardware; uses hardware decoding/encoding to minimise latency【847524538154419†L35-L40】.	README
Pre/post recording	A ring buffer captures seconds of video before and after an event【847524538154419†L37-L38】; clip length is configurable【176380815276105†L19-L37】.	README & Config
Web UI	A Flask server provides a simple dashboard to view live detections, saved clips and action buttons	README & UI

Feature	Description	Citation
	【847524538154419†L39-L40】 【807561843126669†L16-L44】.	
Human-in-the-loop	Operators can acknowledge, dismiss or trigger escalation; auto-escalation is optional when confidence is high 【847524538154419†L16-L20】.	README

Implementation Details

Configuration

The behaviour of I-Guard is controlled by a YAML configuration file (config.yaml). Users define one or more cameras by specifying an identifier, a friendly name, the source (webcam index, file path or RTSP URL), frame rate and resolution 【176380815276105†L3-L17】. Detection settings include the path to the Stage 1 model, input size, confidence threshold and list of classes (person, gun, knife) 【176380815276105†L19-L27】. Flags enable or disable heuristics such as pointing, firing, falls, assault and mass-shooter detection 【176380815276105†L27-L35】. The length of the pre- and post-recording buffer is also set here 【176380815276105†L36-L38】. Stage 2 verification can be enabled or disabled and uses a separate model path and threshold 【176380815276105†L40-L47】. The web server host and port, clip storage directory and log retention period are all configurable 【176380815276105†L49-L60】.

Modular Codebase

The repository is organised into clearly delineated modules:

- **detection package.** Contains the FrameDetector (fast per-frame object detector), ClipVerifier (clip-based verifier) and TrackManager (simple tracking/re-identification) classes 【812088143447372†L0-L20】 【639396526726164†L0-L16】 【736319918174470†L1-L14】.
- **pipeline package.** Implements camera adapters, ring buffers and the InferencePipeline orchestrator. The ring buffer stores frames for pre/post extraction, and the event queue decouples detection from verification 【961720882016570†L91-L114】.
- **ui package.** Provides a Flask application with HTML templates and a JavaScript file that polls the /api/events endpoint and updates the dashboard 【471686557143494†L0-L12】 【807561843126669†L1-L44】.

Extensibility

I-Guard is intentionally minimal; it is a skeleton that can be extended in numerous ways. Developers can:

1. **Replace or augment the heuristics.** Implement pose estimation to detect pointing or integrate muzzle flash detection to distinguish shooting from mere weapon display 【812088143447372†L170-L206】.
2. **Integrate advanced verifiers.** Use action recognition networks (e.g. SlowFast, TimeSformer) in Stage 2 to reduce false positives 【639396526726164†L47-L76】.
3. **Add multi-modal inputs.** Incorporate audio analytics (gunshot detection) or sensor fusion to improve recall.
4. **Implement remote notifications.** Extend the Flask server to send alerts via SMS, email or messaging services.
5. **Improve tracking.** Replace the naive nearest-neighbour tracker with a sophisticated algorithm such as DeepSORT or ByteTrack 【736319918174470†L7-L13】.

Setup and Usage

Deploying I-Guard involves the following steps:

1. **Install dependencies.** Create a Python virtual environment, upgrade pip and install the packages listed in requirements.txt 【847524538154419†L72-L80】.

   ```
   python3 -m venv venv
   source venv/bin/activate
   pip install --upgrade pip
   pip install -r requirements.txt
   ```

2. **Configure cameras and models.** Edit config.yaml to add camera sources, adjust detection thresholds, specify the model paths and set pre/post buffer lengths 【176380815276105†L3-L37】. Download or convert your detection models to TensorRT engines and update the configuration accordingly 【847524538154419†L31-L33】.

3. **Launch the application.** Run the main script:

   ```
   python app.py
   ```

   The Flask web interface will be available at http://localhost:5000 【847524538154419†L91-L96】.

4. **Review events.** Open the dashboard in your browser. Recent events appear in a table with time, camera ID, flags, verification score and a video clip. Operators can acknowledge or dismiss each event, and optional auto-escalation can be configured for high-confidence events 【807561843126669†L16-L44】.

Limitations and Future Work

While I-Guard provides a functional starting point, it is not a complete production system. The current heuristics for pointing, firing, falls and assaults are placeholders that always return False【812088143447372†L170-L207】. The ClipVerifier uses a naïve ratio of weapon frames rather than a trained model【639396526726164†L47-L76】. There is no authentication or access control on the web interface, and storage management lacks automatic pruning beyond a simple keep_days setting【176380815276105†L54-L60】.

Future enhancements could include:

- **Robust detection and action classification.** Integrate state-of-the-art object detectors and action classifiers to detect threats accurately under varied conditions.
- **Contextual reasoning.** Incorporate scene understanding (e.g. recognising classrooms vs. outdoors) to reduce false alarms.
- **Scalability and resource management.** Implement GPU resource pooling and batching to handle dozens of cameras efficiently【961720882016570†L11-L15】.
- **Security and privacy.** Add authentication and encryption to the web interface and log sensitive events appropriately.
- **User feedback loop.** Collect operator feedback to retrain detection models and adapt thresholds over time.

Conclusion

I-Guard demonstrates that real-time weapon and violence detection can be achieved on affordable edge devices without cloud dependencies. Its modular design, configurable pipeline and simple web UI make it a valuable foundation for building more sophisticated safety systems. By incorporating accurate models, better heuristics and secure user interfaces, future iterations of I-Guard or similar projects could play a pivotal role in preventing tragedies and protecting vulnerable spaces.

References

The descriptions of the pipeline, features and configuration are based on the I-Guard repository's documentation and source code【847524538154419†L0-L20】【847524538154419†L29-L40】【176380815276105†L19-L37】【812088143447372†L170-L206】【639396526726164†L47-L76】【961720882016570†L91-L114】.