Università di Pisa
Department of Information Engineering

Master's Degree in Artificial Intelligence and Data Engineering

Industrial Applications Project

Video Violence Detection

Authors:
F. Galardi, M. Meazzini, A. Vagnoli

Academic Year 2025–2026

Contents

1 Abstract

Violence detection in video streams represents a critical challenge for public safety applications, particularly in surveillance scenarios characterized by limited supervision and the need for timely intervention. This work presents an end-to-end edge–server system for automated violence detection based on video analysis, designed for deployment in constrained environments such as commercial transport vehicles. The proposed architecture combines a low-cost embedded device for video acquisition with server-side GPU-accelerated inference, enabling near real-time analysis while overcoming the computational limitations of edge hardware.

An extensive experimental evaluation was conducted to analyze the trade-off between detection performance and system responsiveness. In particular, the impact of different frame sampling rates and decision thresholds was investigated with respect to classification metrics and end-to-end latency. Results show that increasing the temporal resolution improves recall and reduces false negatives, which is critical in safety sensitive scenarios, but at the cost of significantly higher communication and inference latency. Intermediate sampling configurations achieve the best compromise, maintaining robust detection performance while keeping latency within practical bounds.

The findings demonstrate that the proposed system is effective, scalable, and well-suited for real-world surveillance applications. Moreover, the experimental analysis highlights the importance of jointly considering accuracy metrics and latency constraints when designing video-based violence detection systems. The modular design of the architecture further enables future extensions, including on-device inference and temporal consistency modeling, to enhance robustness and deployment flexibility.

2 Introduction

Public safety is a major concern in modern urban environments, particularly in contexts characterized by high passenger density and limited direct supervision. Public transportation systems, such as taxis and buses, represent sensitive scenarios where violent or aggressive behaviors may occur and require timely detection. Traditional surveillance approaches often rely on passive video recording or human monitoring, which can be ineffective due to delayed response times, high operational costs, and limited scalability.

Recent advances in embedded systems and artificial intelligence have enabled the development of automated video analysis solutions capable of continuously monitoring environments and detecting critical events in real time. These technologies offer the potential to improve safety by providing early warnings and supporting rapid intervention, without requiring constant human oversight.

This project proposes an automated system for the detection of violent events based on video analysis. The system captures short video clips using a low-cost embedded device equipped with a camera and transmits them to a remote server for processing. A machine learning model analyzes the incoming data to determine whether violent behavior is present, and the results are exposed through a web-based interface that allows human confirmation. While the actions taken following a confirmed event are outside the scope of this work, the proposed system aims to provide a reliable and scalable foundation for enhancing public safety through intelligent video surveillance.

3 Related Works

Human Activity Recognition (HAR) is a well-established research area within computer vision, with extensive applications in video surveillance and security monitoring. Within this field, violence detection has attracted significant attention due to its relevance for public safety in crowded and unstructured environments.

A representative contribution is the work by Cheng et al. (2021) [1], who introduced the RWF-2000 dataset, a large-scale benchmark designed for violence detection in real-world public scenes. Along with the dataset, the authors proposed a lightweight neural network capable of maintaining good performance under challenging conditions such as occlusions, low resolution, and complex backgrounds.

Using the same dataset, Garcia-Cobo and SanMiguel (2023) [2] explored a skeleton-based approach that leverages human pose estimation and change detection techniques. By modeling interactions through structured skeletal representations, their method achieves higher accuracy compared to appearance-based approaches, highlighting the effectiveness of pose-driven features for violence recognition.

Although the literature on violence detection in public spaces is substantial, most existing works focus on offline analysis or fixed-camera surveillance systems. Fewer studies address end-to-end architectures that integrate edge devices, real-time data transmission, server-side inference, and human-in-the-loop validation. This project contributes to this area by proposing a complete system architecture tailored to automated public safety monitoring.

4 System Description

4.1 Client: Raspberry Pi 3 B+

The edge component of the system is implemented on a Raspberry Pi 3 B+ equipped with a camera module. Its role is to acquire video data from the monitored environment and manage its transmission to the server. Video capture is performed using the rpicam-vid utility, which records short clips with predefined resolution, frame rate, and duration. Each clip is associated with a unique identifier generated using UUIDs, ensuring consistency across system components.

After acquisition, the Raspberry Pi communicates with the server via HTTP requests implemented using the requests library. The system supports two operating modes: full video transmission and frame-based transmission. In the latter, frames are locally extracted using ffmpeg at different sampling rates to reduce bandwidth usage and evaluate performance trade-offs. Each request includes metadata such as timestamps, sampling parameters, and device identifiers, enabling precise measurement of capture, transmission, and processing latency.

4.2 Server: PC

The server is responsible for receiving data from edge devices, executing inference, storing results, and providing user interaction capabilities. It is implemented using the FastAPI framework, which enables efficient handling of asynchronous HTTP requests. Cross-Origin Resource Sharing (CORS) is enabled to support interaction with a web-based interface served directly by the same application.

Uploaded videos and frames are temporarily stored on disk and processed sequentially. The server logs predictions, timing information, and metadata in structured JSON files to support later analysis and evaluation. When enabled, processed videos are annotated and transcoded using FFmpeg to ensure compatibility with standard video players. A REST API exposes the system state and the most recent detection result, while a dedicated endpoint allows a human operator to confirm or correct the predicted label through the web interface. This design supports transparency and human-in-the-loop validation.

4.3 Violence Detection Neural Network

The violence detection module runs entirely on the server and is based on a lightweight neural network exported in ONNX format. Inference is executed using the ONNX Runtime with GPU support. Input frames are preprocessed using OpenCV, including resizing, color space conversion, normalization, and tensor reshaping to match the network's expected input format.

The model follows an object-detection approach, producing bounding boxes, confidence scores, and class labels for each frame. A video clip is classified as violent if at least one detection associated with the violence class exceeds a predefined confidence threshold. This decision logic is consistently applied in both full-video and frame-sampling modes. The modular design of the inference pipeline allows the neural network to be replaced or updated without changes to the rest of the system architecture.

The violence detection model is based on a YOLO-based architecture taken by existing open-source implementations (11.170.202 parameters). [3].

5 Experiment Description

5.1 Preliminary Considerations

The first stage of the experimental activity investigated the feasibility of performing violence detection inference directly on the Raspberry Pi 3 B+. Several publicly available neural network models were evaluated with the objective of executing inference on the edge device.

This approach was found to be unsuitable due to two main limitations. First, many models relied on libraries, frameworks, or architectures that are not fully compatible with the Raspberry Pi 3 B+ platform, preventing successful deployment. Second, the limited hardware resources of the device, in particular the constrained RAM capacity, resulted in excessive memory usage and unacceptable inference latency.

Based on these observations, the project focus was shifted to a server-based inference architecture. The subsequent experimental activity therefore concentrates on optimizing network usage and data transmission strategies while performing inference on the server side.

5.2 Network and Latency Optimization

Following the decision to adopt a server-based inference architecture, the experimental activity focused on the analysis and optimization of network load and data transmission requirements. In scenarios where video streams are continuously acquired at the edge and transmitted to a remote server for inference, network bandwidth and latency become critical factors that directly impact the system's responsiveness and scalability.

For this reason, an exploratory study was conducted to evaluate how the performance of the violence detection network varies under different data transmission strategies. In particular, the analysis focused on the relationship between model performance metrics (accuracy, precision, recall and f1 score) and the number of video frames transmitted to the server. Several configurations were evaluated, ranging from the transmission of complete video sequences to different frame sampling strategies with a reduced number of frames.

In addition to model performance metrics, temporal performance metrics were considered, including uploading time and inference time, in order to assess the trade-off between detection performance and system responsiveness. Furthermore, the effect of varying the network confidence threshold was analyzed, with the objective of understanding its impact on detection reliability, false positives, and overall system behavior.

This exploratory analysis allowed us to investigate whether it is possible to optimize network usage while maintaining acceptable detection performance, identifying configurations that reduce data transmission and latency without significantly degrading accuracy.

5.3 Experiment setup

5.3.1 Tools setup

The experimental setup is composed of an embedded edge device and a host computer acting as a server.

The edge device is a Raspberry Pi 3 Model B+, equipped with a Raspberry Pi Camera Module v2 and a 64 GB microSD card for local storage.

- https://www.raspberrypi.com/products/raspberry-pi-3-model-b/

Figure 1: The Raspberry Pi equipped with the Camera Module v2 is powered via an external power supply and connected to the internet through an Ethernet connection.

The host computer, which acts as the server for data processing and model execution, has the following hardware specifications:

- Processor: 11th Gen Intel(R) Core(TM) i7-1195G7 @ 2.90 GHz
- Installed RAM: 16.0 GB
- System type: 64-bit operating system, x64-based processor
- Graphics card: Intel(R) Iris(R) Xe Graphics

Neural network inference is performed on the host computer using GPU acceleration provided by the integrated Intel Iris Xe Graphics.

During the experiments, communication between the Raspberry Pi and the host computer is established via SSH. Both devices are connected to the same local network: the host computer is connected to the router via Wi-Fi, while the Raspberry Pi is connected via Ethernet. This configuration ensures reliable and low-latency data exchange throughout the experimental process.

5.3.2 Environment setup

The experimental environment was designed to emulate the interior conditions of a commercial transport vehicle with two passengers.

The Raspberry Pi equipped with the camera module was positioned at a height of 80 cm from the ground, ensuring a frontal viewpoint comparable to an onboard monitoring system. In front of the camera, two chairs were placed side by side at a distance of 150 cm from the Raspberry Pi. This arrangement was used to simulate the presence and relative positioning of two people seated inside a commercial transport vehicle.

Figure 2: Real experimental environment setup simulating two passengers inside a commercial transport vehicle.

5.3.3 Experiment and dataset creation

The experiment was conducted with the dual objective of evaluating the system performance and creating a dedicated dataset under controlled yet realistic conditions.

A total of 60 video recordings were collected, equally divided into two classes: 30 *Violence* videos and 30 *NoViolence* videos. For each recording, predictions were performed at different frame rates (FPS) and using multiple decision thresholds. During this process, system performance metrics were collected, including inference latency and uploading latency, in order to analyze the impact of FPS and threshold selection on both accuracy and real-time responsiveness.

Two participants took part in the experiment. They alternated their positions on the two chairs to avoid positional bias and to increase variability within the dataset. Recordings were performed both with single-person scenarios and with two people simultaneously present, enabling the dataset to reflect a range of realistic interaction conditions typical of a commercial transport vehicle environment.

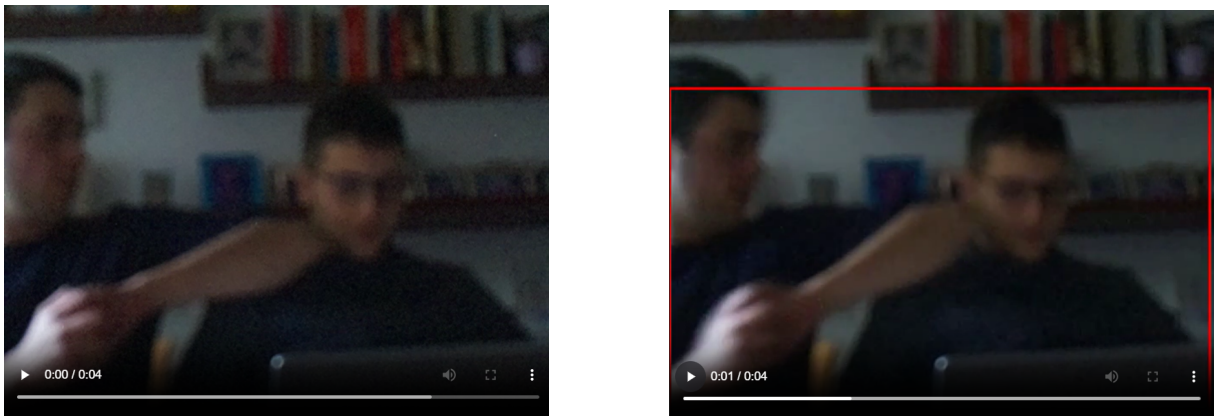


Figure 3: Example frames from the experiment: the left image shows one frame of the input video stream, while the right image shows one frame of the output video with the red bounding box highlighting a frame classified as violent.

5.3.4 Experiment records and logged parameters

During the experimental campaign, each recording generated a structured log entry containing metadata, inference outputs, and timing measurements. These records were used both for performance evaluation and for the analysis of system behavior under different operating conditions.

Experiments were conducted using two processing modes: *video*-based inference and *frame*-based inference. For the frame-based mode, fixed sampling frame rates were used, namely 1 FPS, 2 FPS, and 5 FPS, while the video-based mode operated at a fixed acquisition rate of 10 FPS. For each recording, predictions were evaluated across multiple decision thresholds.

Table 1 summarizes the fields included in each experimental record.

Field	**Description**
clip_id	Unique identifier associated with each recorded video sequence.
mode	Inference modality used during the experiment (*video* or *frames*).
device_id	Identifier of the acquisition device (Raspberry Pi with Camera Module v2).
sampling_fps	Frame sampling rate used for inference (1, 2, 5 FPS for frames; 10 FPS for video).
num_frames	Number of frames processed during inference.
prediction	Final classification output of the model (Violence / NoViolence).
predictions_by_threshold	Model predictions evaluated at different decision thresholds.
ground_truth	Ground-truth label associated with the recording.
is_correct	Boolean flag indicating whether the final prediction matches the ground truth.
capture / sampling time	Time required to acquire or sample frames on the Raspberry Pi.
inference time	Time required by the server to perform neural network inference.
upload_client time	End-to-end upload latency measured from the Raspberry Pi, starting from data transmission initiation to the reception of the acknowledgment (ACK) from the server.

Table 1: Description of the parameters and measurements stored for each experimental record.

6 Results and Considerations

6.1 Results Analysis

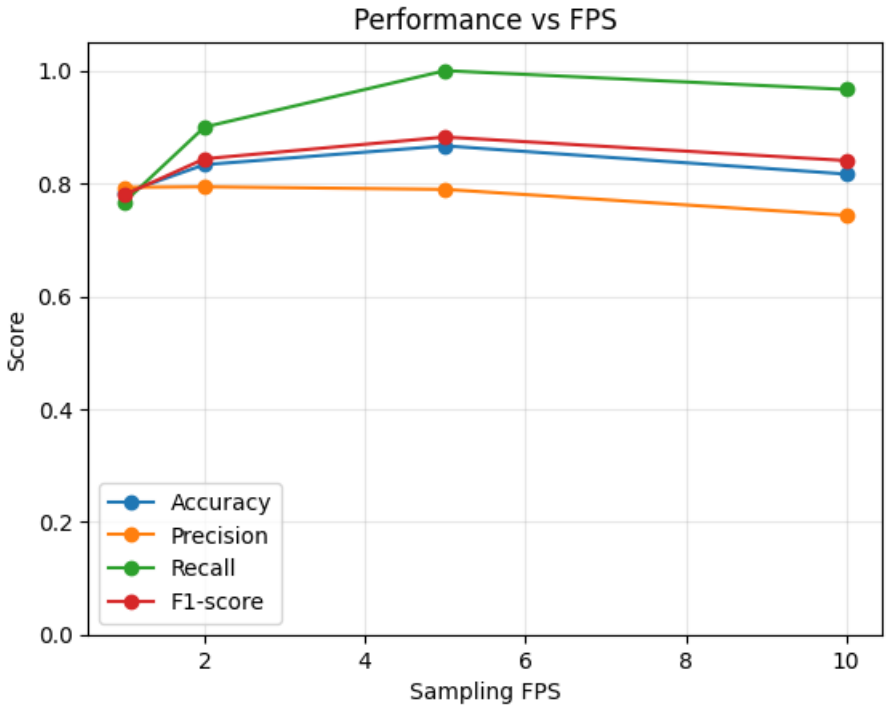
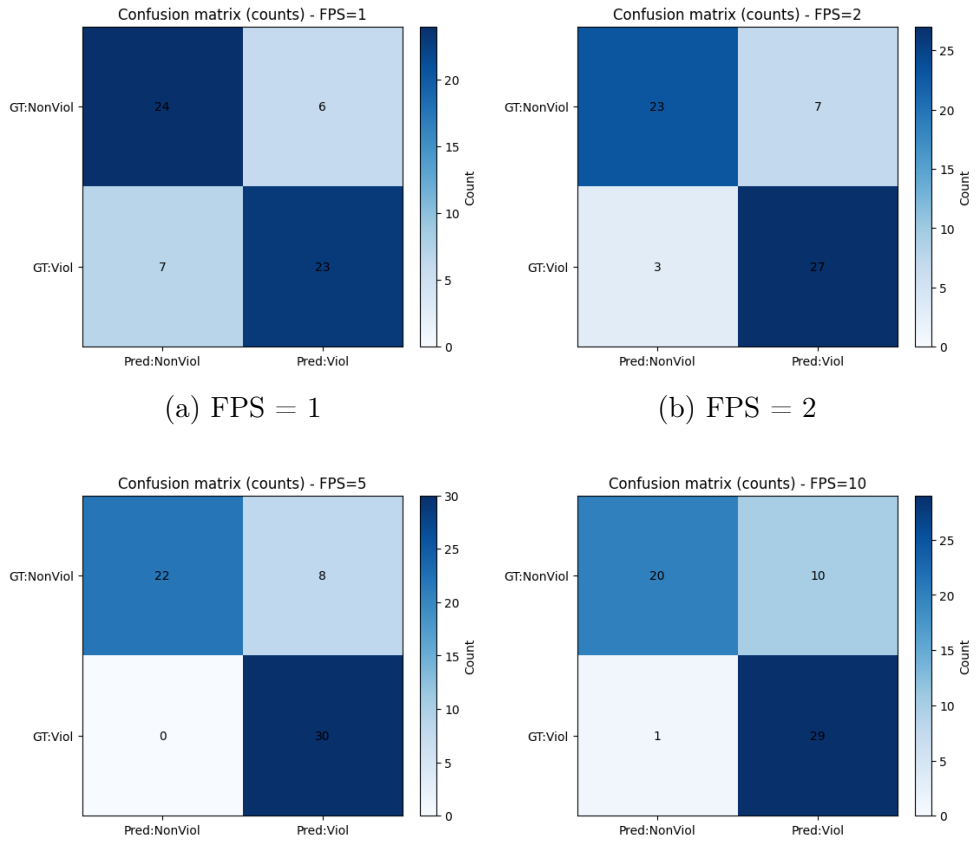


Figure 4: Classification performance (accuracy, precision, recall) as a function of the sampling frame rate.

The figure shows the relationship between sampling frame rate (FPS) and classification performance in terms of accuracy, precision, and recall. As the FPS increases, recall exhibits a clear improvement, reaching its maximum at intermediate frame rates and remaining consistently high even at higher FPS values. This indicates that the system becomes increasingly effective at correctly identifying violent events when more temporal information is available.

While accuracy and precision show moderate variations across different FPS settings, recall remains the most critical metric in the context of violence detection. In safety-critical surveillance scenarios, minimizing false negatives is of primary importance, as failing to detect a violent event may have significantly more severe consequences than generating false positives. From this perspective, configurations that favor higher recall are preferable, even at the cost of a slight reduction in precision.

Overall, the results suggest that increasing the sampling FPS improves the system's ability to capture relevant temporal cues associated with violent behavior, thereby reducing the likelihood of missed detections and enhancing the reliability of the violence detection pipeline.



(a) FPS = 1 (b) FPS = 2

(c) FPS = 5 (d) FPS = 10

Figure 5: Confusion matrices (absolute counts) for different sampling frame rates.

The confusion matrices reported for different sampling frame rates highlight the impact of FPS on the system's ability to correctly detect violent events. At lower frame rates (FPS = 1), the model exhibits a non-negligible number of false negatives, indicating that some violent sequences are misclassified as non-violent due to limited temporal information. This behavior is critical in violence detection scenarios, where missed detections directly reduce system reliability.

As the sampling rate increases to FPS = 2, a clear reduction in false negatives can be observed, accompanied by an improvement in true positive detections. This suggests that even a modest increase in temporal resolution significantly enhances the model's capability to capture motion patterns associated with violent behavior.

At FPS = 5, the system achieves the best balance for violence detection, completely eliminating false negatives in the evaluated dataset. All violent events are correctly classified, demonstrating that this frame rate provides sufficient temporal context for robust detection. Although a small increase in false positives is observed, this trade-off is acceptable in safety-critical applications where prioritizing recall is essential.

Finally, at FPS = 10, the system maintains a very low number of false negatives while slightly increasing false positives. This indicates diminishing returns in recall improvement compared to FPS = 5, while introducing additional computational and communication overhead.

Overall, these results confirm that increasing the sampling frame rate substantially reduces false negatives and improves recall. In the context of violence detection, where avoiding missed violent events is more important than minimizing false alarms, intermediate-

to-high FPS configurations represent the most suitable operating point for real-world deployment.

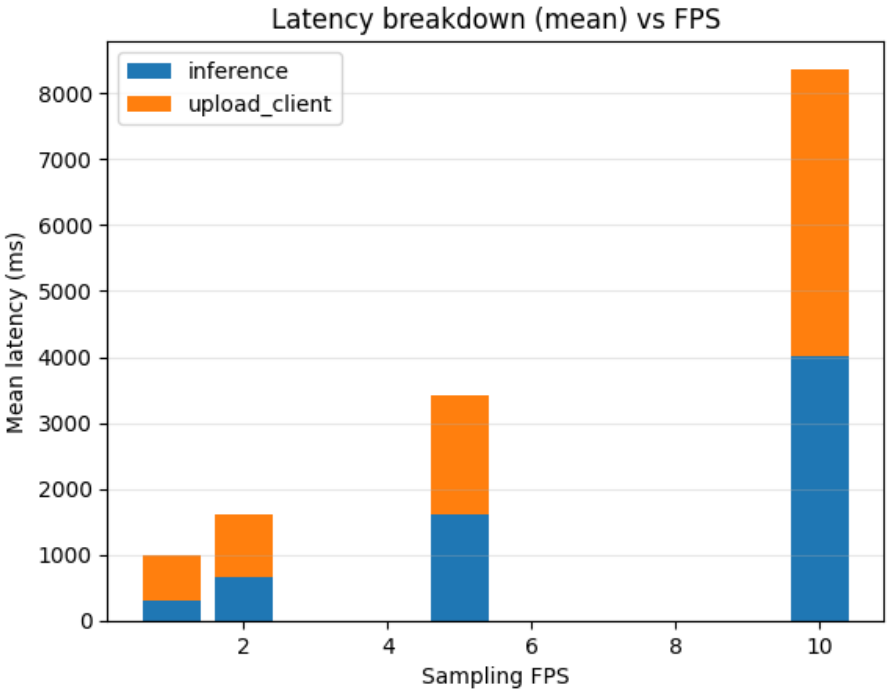


Figure 6: Average latency breakdown by component (inference and upload) for different sampling frame rates.

The figure presents the mean latency breakdown as a function of the sampling frame rate, separating inference time from client-side upload latency. As the FPS increases, both components contribute significantly to the overall latency growth, but with different scaling behaviors.

At lower frame rates, the total latency is dominated by communication overhead, with the upload client time representing a substantial portion of the end-to-end delay. This highlights the impact of network transmission and synchronization costs even when the computational workload is relatively low.

As the FPS increases, inference latency becomes progressively more dominant. The higher number of frames processed per unit time leads to increased GPU utilization and longer inference queues on the server, resulting in a steep rise in inference-related delays. At the highest FPS setting, inference clearly represents the primary bottleneck of the system.

These results indicate that the system is constrained by both communication and computation, depending on the operating point. At low FPS, optimizing data transmission and reducing upload overhead would yield the largest benefits, whereas at higher FPS, improvements in model efficiency or hardware acceleration would be required to sustain real-time performance.

Overall, the latency breakdown reinforces the need for a balanced configuration that considers both detection performance and system constraints. Intermediate FPS values offer a favorable compromise, limiting inference overload while maintaining acceptable communication latency and robust violence detection capabilities.

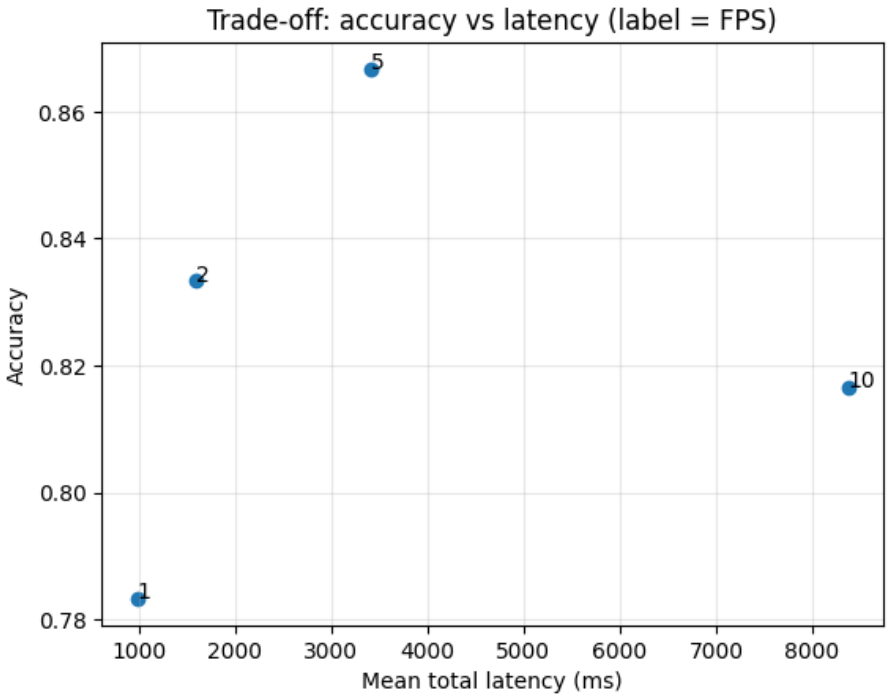


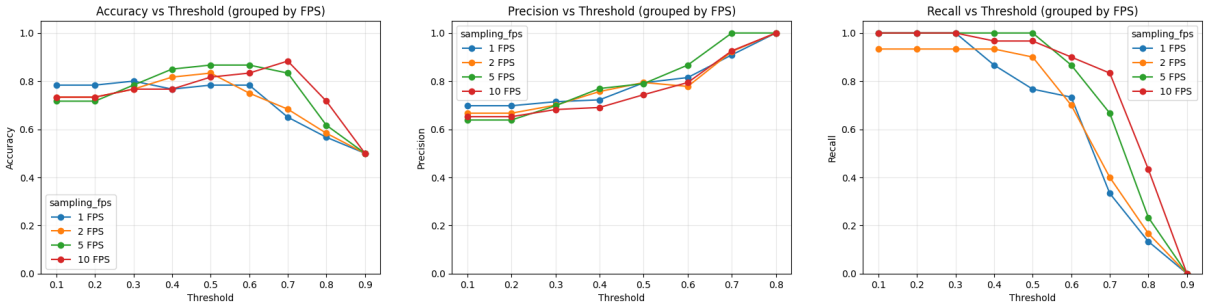
Figure 7: Trade-off between classification accuracy and mean end-to-end latency, with labels indicating the sampling frame rate.

The figure illustrates the trade-off between classification accuracy and mean end-to-end latency, with each point corresponding to a different sampling frame rate. As expected, increasing FPS generally improves accuracy by providing richer temporal information, but at the cost of substantially higher latency.

At very low frame rates, the system achieves minimal latency but suffers from reduced accuracy, making it less reliable for violence detection. Conversely, the highest FPS configuration leads to the largest latency while offering unsatisfying accuracy levels.

The intermediate FPS setting represents the most favorable operating point, achieving the highest accuracy while maintaining a moderate latency. This configuration effectively balances detection performance and system responsiveness, aligning well with the requirements of real-time or near real-time surveillance applications.

From a practical perspective, this trade-off analysis confirms that selecting an intermediate frame rate enables robust violence detection while avoiding excessive computational and communication overhead.



(a) Accuracy (b) Precision (c) Recall

Figure 8: Classification metrics as a function of the decision threshold, grouped by sampling frame rate.

The figures analyze the impact of the decision threshold on accuracy, precision, and recall for different sampling frame rates. As expected, the threshold plays a critical role in balancing detection sensitivity and classification reliability across all FPS configurations.

From the accuracy perspective, intermediate threshold values generally lead to the best performance, while very high thresholds cause a noticeable degradation. This behavior is particularly evident at lower FPS values, where limited temporal information makes the model more sensitive to overly conservative decision boundaries.

Precision consistently increases with higher thresholds for all FPS settings. This trend reflects a reduction in false positives, as the model becomes more selective in declaring violent events. However, this improvement in precision comes at the cost of recall, especially at higher thresholds.

Recall exhibits the most critical behavior for violence detection. At low and intermediate thresholds, recall remains close to its maximum across all FPS configurations, indicating a strong ability to correctly identify violent events. As the threshold increases beyond moderate values, recall drops sharply, leading to a significant rise in false negatives. This effect is particularly pronounced for lower FPS values, whereas higher FPS configurations demonstrate greater robustness and maintain higher recall for a wider range of thresholds.

Overall, these results confirm the inherent trade-off between precision and recall controlled by the decision threshold. In the context of violence detection, maintaining a high recall is essential to minimize missed violent events, even if this leads to a moderate increase in false positives.

6.2 Summary of the results

Combining the results from performance metrics, confusion matrices, latency analysis, and threshold evaluation, a clear and consistent operating point emerges for the proposed system.

Low FPS configurations provide minimal latency but suffer from reduced recall and higher sensitivity to threshold selection, increasing the risk of false negatives. Conversely, high FPS configurations improve recall robustness but introduce substantial end-to-end latency and computational overhead, with diminishing gains in overall accuracy.

Intermediate FPS settings achieve the best compromise between detection performance and system responsiveness. In particular, this configuration maximizes recall, eliminates or strongly reduces false negatives, and maintains acceptable accuracy while keeping la-

tency within practical limits for near real-time applications. Additionally, intermediate decision thresholds allow the system to preserve high recall without excessively sacrificing precision.

Based on the experimental evidence, the most effective configuration of the network is obtained by combining:

- a sampling frame rate of 5 FPS,
- a decision threshold of 0.5

This configuration ensures reliable violence detection with minimal missed events, balanced precision, and sustainable latency, making it well-suited for deployment in real-world surveillance scenarios such as commercial transport vehicles.

7 Project Demo

The project demo was designed to showcase the complete end-to-end pipeline of the proposed system, from video acquisition to violence detection inference, using a simplified and controlled setup for demonstrative purposes.

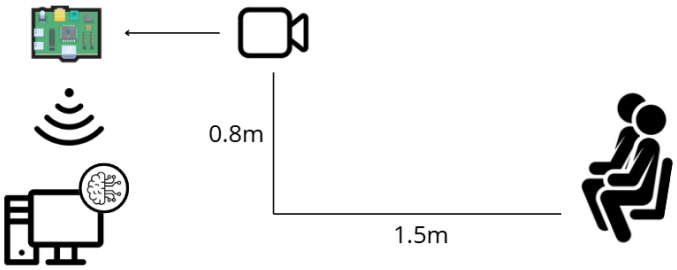


Figure 9: System setup used for the demo.

Figure 9 illustrates the system configuration adopted during the demo. A camera is connected to a Raspberry Pi device, which is responsible for video acquisition. The Raspberry Pi is connected via Ethernet to a Wi-Fi router, enabling network communication with a remote server running on a personal computer. The server hosts the neural network model and performs the inference on the received video data.

The demo was intentionally structured to process one video clip at a time. This design choice was made to simplify the demonstration and clearly highlight each stage of the pipeline.

The demonstration procedure starts by powering the Raspberry Pi and connecting it to the network. Once the device is online, an SSH connection is established from the PC in order to deploy and manage the execution of the acquisition script on the Raspberry Pi. Subsequently, the inference server is launched on the PC by executing the `app.py` script, which initializes the backend services and the web-based user interface.

After the server is running, the acquisition process is started remotely via SSH by executing the `raspberry.py` script on the Raspberry Pi. This script triggers the recording of a video clip with a fixed duration of four seconds. Once the recording is completed, the clip is automatically transmitted to the server, where it is processed by the neural network model to perform violence detection inference.

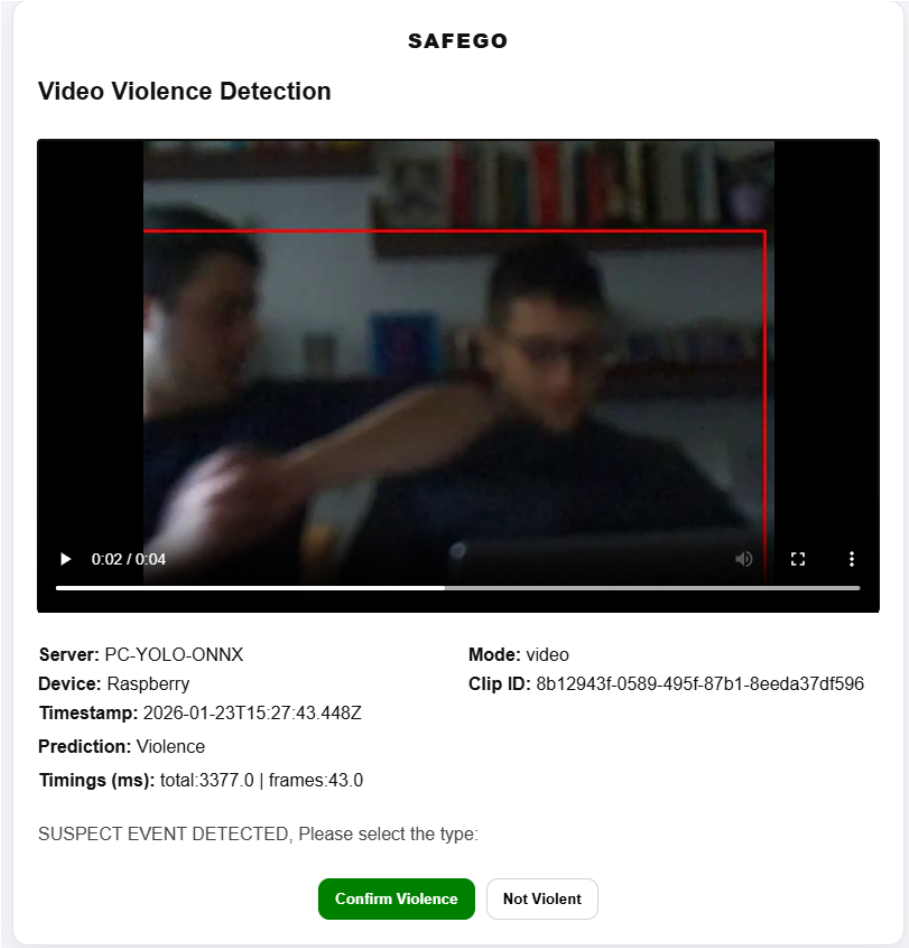


Figure 10: UI of the server with the violence detected video examples recorded and sent by raspberry

The recorded clip and the corresponding inference results are then displayed through the web-based user interface. In the presence of detected violent behavior, red bounding boxes show the individuals involved in the scene. Additionally, when violence is detected, the user interface presents confirmation buttons that allow an operator to validate the detection. This interaction step was introduced to mitigate potential issues related to false positives and to emphasize the role of human-in-the-loop supervision in practical deployments.

8 Conclusions and Future Work

8.1 Conclusions

This work presented an edge–cloud violence detection system designed for surveillance scenarios in commercial transport environments. The proposed architecture combines an embedded acquisition device with server-side GPU-accelerated inference, enabling real-time or near real-time analysis while maintaining flexibility in system configuration.

Through an extensive experimental evaluation, we analyzed the impact of sampling frame rate and decision threshold on detection performance and system latency. The results highlighted a clear trade-off between accuracy, recall, and end-to-end latency. In particular, intermediate sampling frame rates achieved the most favorable balance, significantly reducing false negatives while keeping latency within acceptable bounds. This is especially important in violence detection applications, where missed detections may have severe consequences. The threshold analysis further confirmed that moderate decision thresholds allow the system to maintain high recall without excessively increasing false positives.

Overall, the experimental findings demonstrate that the proposed system is effective, robust, and suitable for deployment in real-world monitoring scenarios, providing reliable violence detection under realistic operating conditions.

8.2 Future work

Several directions can be explored to further improve the system. First, future experiments will include longer and more diverse datasets, as well as recordings captured using higher-quality cameras, in order to evaluate system robustness under more challenging visual conditions.

Second, the adoption of more powerful embedded platforms, such as newer Raspberry Pi models or alternative edge devices, could enable on-device inference. Performing inference directly on the edge would significantly reduce communication latency and improve privacy, as raw video data would no longer need to be transmitted to a remote server.

Finally, future work will focus on extending the decision logic beyond single-frame predictions. Instead of classifying violence based on isolated frames, temporal consistency can be introduced by detecting violent events only when multiple consecutive frames exceed the decision threshold. This approach is expected to improve robustness, reduce spurious detections, and better capture the temporal dynamics of violent behavior.

These developments would further enhance the practicality, privacy awareness, and reliability of the proposed violence detection system.

9 Code

The source code used for the implementation of the proposed system is publicly available. The repository includes the complete implementation of the violence detection pipeline, the trained model, the video DEMO and detailed instructions for setting up the required software environment and libraries.

The code can be accessed at the following link:

- `https://github.com/AndreaVGN/video-violence-detection`

Due to privacy considerations, the video recordings used to create the experimental dataset are not publicly released. However, they can be made available upon reasonable request for research and evaluation purposes.

References

[1] M. Cheng, K. Cai, and M. Li, "Rwf-2000: An open large scale video database for violence detection," in *Proceedings of the 25th International Conference on Pattern Recognition (ICPR)*, pp. 4183–4190, 2021.

[2] G. Garcia-Cobo and J. C. SanMiguel, "Human skeletons and change detection for efficient violence detection in surveillance videos," *Computer Vision and Image Understanding*, vol. 233, p. 103739, 2023.

[3] Musawer, "Fight and violence detection using yolov8." `https://github.com/Musawer1214/Fight-Violence-detection-yolov8`, 2023. Accessed: Jan. 2026.