HOME SURVEILLANCE IN GENERAL

BY

GUE KAI KIT

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

FEBRUARY 2025

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Prof. Dr. Leung Kar Hang, for providing me with the opportunity to undertake this project on advanced home surveillance systems. Your expertise, guidance, and support throughout this journey have been invaluable, and I am deeply grateful for the knowledge and skills I have gained under your mentorship. A million thanks to you.

Finally, I would like to extend my heartfelt thanks to my parents and my family for their unwavering love, support, and continuous encouragement throughout this project. Your belief in me has been my greatest source of strength, and I am truly thankful for everything you have done to help me reach this milestone.

ABSTRACT

This project addresses a growing concern in modern society – home security. With increasing incidents of property crime and unauthorized intrusions, there is a rising demand for intelligent surveillance systems that go beyond the limitations of conventional CCTV setups, which often struggle with false alarms and require manual supervision. This project proposes a smart home surveillance system that combines real-time object detection with violence recognition by leveraging state-of-the-art deep learning techniques.

The system uses the YOLO (You Only Look Once) framework to detect the presence of weapons, offering rapid identification of potential threats. Simultaneously, a ResNet50-based Convolutional Neural Network (CNN) combined with a Long Short-Term Memory (LSTM) network is employed to recognize violent actions over time, such as assaults or robberies, using temporal video frame analysis. When a human is detected in the scene, these detection modules are triggered to identify weapons or violent movements. If a threat is confirmed, the system issues an immediate alert to property owners or security personnel, enabling quick intervention.

By integrating real-time weapon and violence detection in a multi-threaded monitoring system, this solution enhances home surveillance effectiveness and responsiveness, aiming to create a safer and smarter living environment.

Area of Study (Minimum 1 and Maximum 2): Computer Vision, Smart Surveillance Systems

Keywords (Minimum 5 and Maximum 10): Real-time Object Detection, Violence Detection, ResNet50-LSTM, YOLO Algorithm, Home Surveillance, Intrusion Detection, Weapon Detection, Smart Security System, Deep Learning

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CCTV	Closed-Circuit Television
YOLO	You Only Look Once
ResNet50	Residual Network
LSTM	Long short-term memory
ROI	Region of Interest
MIL	Multiple Instance Learning
AUC	Area under Curve
EER	Equal Error Rate
IoT	Internet of Things
SHE	Smart Home Environment
AI	Artificial Intelligence
DL	Deep learning
DCT	Discrete Cosine Transform
HMM	Hidden Markov Model
2s-AGCN	Two-stream Adaptive Graph Convolutional Network
SGD	Stochastic Gradient Descent

Chapter 1
Introduction

1.1 Problem Statement and Motivation

Video surveillance, commonly known as CCTV (closed-circuit television), remains one of the most prevalent methods for home security. However, the development of an effective home surveillance system is crucial to address the significant limitations of current methods, which are plagued by human error, vigilance decrement, lack of real-time alert integration, and high false alarm rates. Security personnel or homeowners monitoring CCTV footage often suffer from vigilance decrement, where their ability to maintain attention and detect unusual activities decreases over time. This decline in vigilance is particularly problematic during long monitoring periods or when critical events occur infrequently, leading to reduced detection reliability, missed detections, and delayed responses [1]. Furthermore, many traditional home surveillance systems are not integrated with real-time alert mechanisms, meaning that even if suspicious activity is recorded, it may not be detected or acted upon promptly. These systems also frequently generate a high rate of false alarms, causing unnecessary disruptions and reducing the reliability and trustworthiness of the security measures. Therefore, there is a pressing need to develop a sophisticated home surveillance system that leverages advanced technologies to minimize human error, ensure continuous and reliable monitoring, integrate real-time alert mechanisms, and reduce false alarms. Such a system would enhance overall home security and provide timely responses to potential threats.

The motivation for developing an advanced home surveillance system lies in addressing these significant limitations to enhance the safety and security of residential properties. Traditional home security measures often fail to provide timely and reliable detection of burglaries, resulting in increased vulnerability and significant financial and emotional distress for homeowners. Moreover, the absence of real-time alert integration in existing systems results in delayed responses to potential threats, exacerbating the risk to residents and their properties. Implementing a system that provides real-time alerts and automated responses is crucial for timely intervention and effective security management in a home environment. Additionally, the underreporting of burglaries and incomplete surveillance footage analysis hinder accurate assessments of burglary trends and the effectiveness of security measures. The proposed system aims to ensure continuous, reliable detection, timely response, and comprehensive data analysis, ultimately reducing the financial and emotional impact of home intrusion and providing greater peace of mind to homeowners.

1.2 Objectives

The primary objective of this project is to develop an intelligent, real-time home surveillance system that minimizes the need for constant human supervision while addressing the limitations of traditional CCTV systems—particularly their high false alarm rates and lack of contextual threat understanding. The system will automatically detect human presence in live video feeds and further analyze the identified individuals to recognize high-risk behaviors, such as violent motions and the presence of weapons, which may indicate potential intrusions or threats.

To enhance situational awareness and responsiveness, the system will incorporate a real-time, on-screen alert mechanism that activates only when predefined threat conditions are met. Each alert will clearly display the event type and associated confidence level, enabling rapid verification without the need for continuous monitoring. Additionally, the system will log all detected events for later review and evidence collection.

By leveraging advanced deep learning architectures—specifically YOLO for weapon detection and a ResNet50 (Residual Network) + LSTM (Long short-term memory) model for violence recognition—along with multithreaded processing, the system aims to achieve high detection accuracy, low latency, and minimal false positives. This ensures that security alerts are both timely and reliable, allowing users to respond promptly to genuine threats.

This project focuses exclusively on the development of software-based threat detection and alerting components. It does not include the implementation of physical deterrents (e.g., sirens, automated locks) or direct integration with external security hardware.

1.3 Project Scope and Direction

At the end of this project, we aim to deliver an intelligent home surveillance system that addresses key limitations of conventional CCTV setups, such as lack of real-time threat detection and reliance on manual monitoring. The system will focus on detecting human presence within user-defined regions and identifying suspicious or dangerous behaviors, specifically violence and the presence of weapons.

To achieve this, the system will leverage a combination of deep learning and computer vision techniques. Human and weapon detection will be performed using the YOLO (You Only Look Once) object detection algorithm, while violent actions will be identified through a ResNet50 + LSTM-based model, which processes temporal patterns in video footage. These components will operate in real-time within a multi-threaded architecture, ensuring efficient and accurate detection without compromising performance.

Upon identification of a potential threat—be it the presence of a weapon or violent behavior, the system will immediately activate an on-screen alert system. Each alert will display the type of threat detected along with its associated confidence score, providing the user with clear, actionable information. Simultaneously, detailed event logs—including timestamps, frame captures, and classification outcomes—will be recorded for post-incident review and evidence collection.
The system will be trained and evaluated using relevant subsets of the UCF-Crime dataset, which contains diverse real-world crime scenes. This will ensure that the model is exposed to realistic scenarios for improved generalization and reliability in home security applications.

1.4 Contributions

The proposed home surveillance system offers significant benefits by enhancing the accuracy and responsiveness of security measures, thereby reducing the prevalence of home intrusions and associated losses. This project has the potential to enhance home security, providing a reliable and proactive solution to a pervasive problem. The advanced detection capabilities will not only protect valuables but also restore the sense of safety and peace of mind for homeowners and communities. Furthermore, the system's scalability and adaptability make it suitable for various applications, from private homes to residential complexes, thereby broadening its impact and utility.

Moreover, the system reduces the need for human resources in monitoring, as it can automatically detect intrusions without requiring constant human oversight. This reduces errors associated with human fatigue and attention lapses, leading to more effective and efficient security management. By leveraging advanced technologies, the project contributes to the development of smarter, more resilient home security systems that can adapt to evolving threats and enhance overall residential safety.

1.5 Report Organization

This report is organized into seven chapters, each addressing different aspects of the intelligent home surveillance system developed in this project. Chapter 1 introduces the project by outlining the problem statement, objectives, scope, key contributions, and the overall structure of the report. It highlights the motivation behind developing an advanced surveillance solution to overcome the limitations of traditional CCTV systems, such as high false alarm rates and lack of contextual understanding.

Chapter 2 presents a literature review on existing smart surveillance solutions and related technologies. It analyzes current methods and tools used for anomaly detection, weapon recognition, and violence detection in video surveillance, identifying their strengths and limitations. This review establishes the need for an integrated, deep learning-based approach to improve home security systems.

Chapter 3 describes the system methodology, including the design architecture, use case diagram, and activity diagram. It provides an overview of how the system processes video input, detects person using YOLOv8 within a user-defined region of interest (ROI), and applies ResNet50+LSTM for violence recognition and another YOLOv8 model for weapon detection. The threaded implementation of detection modules and real-time alert handling are also discussed.

Chapter 4 explains the system design in detail, covering the block diagram, specifications of software components used (such as OpenCV, PyTorch, and threading libraries), and the interactions between components during the detection and alert process. This chapter emphasizes the flow from human detection to violence and weapon detection, culminating in real-time on-screen alerts.

Chapter 5 focuses on system implementation, describing how the models were implemented and integrated, how the software environment was set up, and how the system operates in real-time. It includes screenshots, explanations of system behavior, and discusses issues encountered during development, such as frame processing delays and model optimization challenges.

Chapter 6 evaluates the system’s performance through testing using curated datasets. It presents the evaluation metrics, testing setup, results for detection accuracy and alert precision, discusses challenges such as false positives and missed detections, and reflects on how well the objectives were achieved.

Finally, Chapter 7 concludes the report by summarizing the achievements and limitations of the project. It also provides recommendations for future work, such as improving model accuracy through larger and more diverse datasets, enhancing system robustness in real-world conditions, optimizing UI responsiveness, and integrating the system with IoT-based physical alert mechanisms like sirens or smart notifications via external devices.

Chapter 2

Literature Review

Previous works on Anomaly Detection and Human Action Recognition

2.1 Real-world Anomaly Detection in Surveillance Videos

Sultani et al. [2] presented a novel deep learning architecture for real-world anomaly detection in surveillance videos. The proposed model considers normal and anomalous videos as bags and video segments as instances in multiple instance learning (MIL) and automatically learn a deep anomaly ranking model that predicts high anomaly scores for anomalous video segments. It incorporates spatial and temporal information by employing both 2D CNNs for spatial feature extraction and 3D CNNs for temporal feature modelling. This architecture enables the model to capture complex spatio-temporal patterns inherent in surveillance videos.

The dataset consists of 1900 long and untrimmed real-world surveillance videos with a duration of 128 hours, with 13 realistic anomalous activities. The surveillance videos would be divided into segments and are used to detect anomaly and differentiate each anomaly.

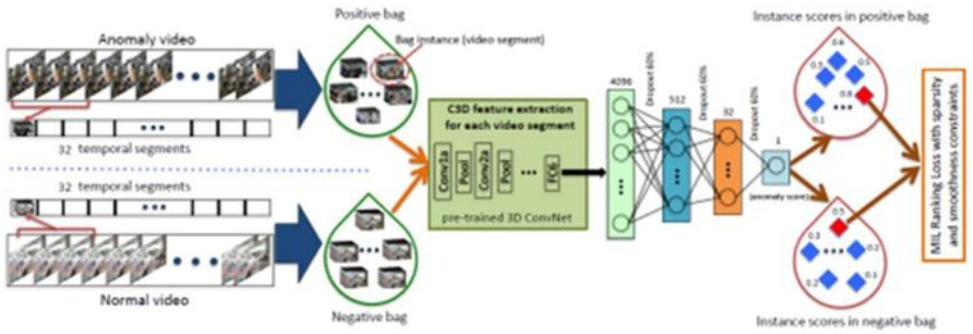

Anomaly video
32 temporal segments
32 temporal segments
Normal video
Positive bag
Bag instance (video segment)
Negative bag
C3D feature extraction for each video segment
Conv1a
Pool
Conv2a
Pool
FC6
pre-trained 3D ConvNet
4096
Dropout 60%
512
Dropout 60%
32
Dropout 60%
1
(anomaly score)
Instance scores in positive bag
Instance scores in negative bag
MIL Ranking Loss with sparsity and smoothness constraints

Figure 2.1 The flow diagram of the proposed anomaly detection approach.

As seen from the above flowchart, the divided segments would be use as bag instances and proceed to 3D convolution features extraction. These features are then used to train a deep MIL Ranking Model which computes the ranking loss between the highest scores instances in the positive and negative bag.

Given a lot of positive and negative videos with video-level labels, the network can automatically learn to predict the location of the anomaly in the video. This could be done by training the network to produce high scores for anomalous video segments during training iterations. Although we do not use any segment level annotations, the network is able to predict the temporal location of an anomaly in terms of anomaly scores.

The area under curve (AUC) is used to evaluate the performance instead of equal error rate (EER) as EER does not measure anomaly correctly. Results show that this approach has the highest AUC of 75.41, as compared to deep auto-encoder based approach and dictionary based approach with AUC of 50.6 and 65.51 respectively. In addition, this approach also has the lowest false alarm rate of 1.9, indicating a more robust anomaly detection system. The false alarm rate of the deep auto-encoder based approach and dictionary based approach are 27.2 and 3.1 respectively.

However, there are a few limitations to this approach. The system would fail to identify anomaly if the environment is dark. This would not be practical as anomaly usually happens at night and the system is preferred to be active at all times. Secondly, it would generate false alarm if there were occlusions such as flying insects or sudden people gathering, potentially leading to unnecessary disruptions and reduced system reliability. Implementing advanced object tracking algorithms capable of distinguishing between genuine threats and transient occlusions can mitigate this challenge.

The model proposed by Sultani et al. excels in controlled environments, achieving high accuracy and low false alarm rates. However, its effectiveness diminishes under low-light conditions and when faced with occlusions. The reliance on multiple instance learning without segment-level annotations is innovative but limits its application in real-world scenarios where lighting and occlusions are variable. For our home surveillance system, integrating advanced object tracking algorithms and ensuring consistent performance across diverse environments can enhance the reliability of anomaly detection. Furthermore, the approach of using MIL can be adapted to distinguish between normal and suspicious activities in home surveillance feeds, provided the dataset includes a diverse range of lighting conditions and potential occlusions.

2.2 IoT-Guard: Event-Driven Fog-Based Video Surveillance System for Real- Time Security Management

Sultana et al.[3] proposed the IoT-Guard, a distributed Internet of Things (IoT) framework designed for intelligent, resource-efficient and real-time security management in a smart home environment (SHE). The system incorporates edge-fog computational layers to aid in crime prevention and predict crime events. Utilizing Artificial Intelligence (AI) and an event-driven approach, IoT-Guard detects and confirms crime events in real-time, facilitating immediate action by protective services and police units while conserving resources such as energy, bandwidth, and memory usage.

Unlike traditional approaches, IoT-Guard employs an event-driven strategy, where surveillance data is forwarded to fog nodes only when specific events (e.g., motion) are detected, reducing energy consumption and bandwidth usage. Edge computing enables this approach by delegating processing to camera-connected IoT-edge-node devices, while fog computing enables AI integration to make automated decisions based on gathered information. Deep learning (DL) algorithms enhance AI capabilities, allowing fog nodes to predict events and take proactive measures to prevent crime incidents in smart home environment (SHE).

The security management framework of the proposed IoT-Guard is described, focusing on crime detection and proactive alerts using edge-and fog-integrated approaches. Fig. 2.2 depicts the high-level architecture, featuring camera-connected edge nodes covering residential units. An event-driven feature keeps edge nodes on standby until significant movement is detected, upon which they capture and forward motion-detected images to fog nodes. Each fog node controls several edge nodes within a unit or building, with multiple fog nodes covering entire residential areas.

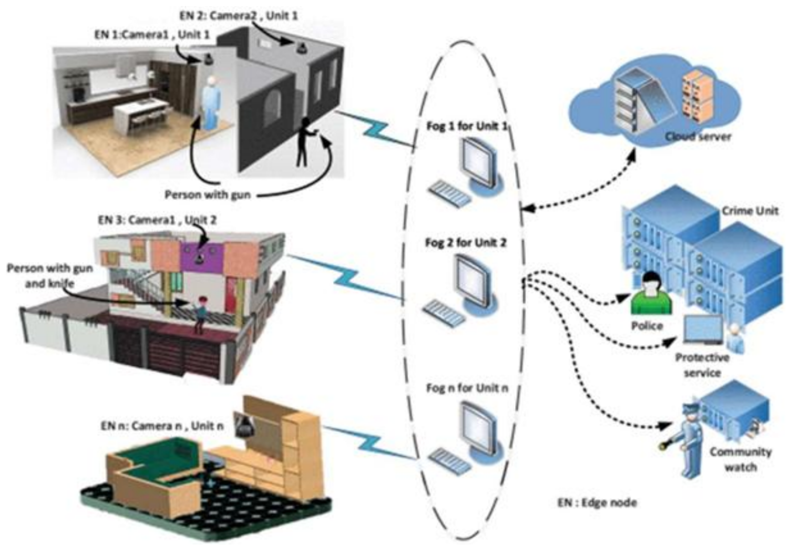
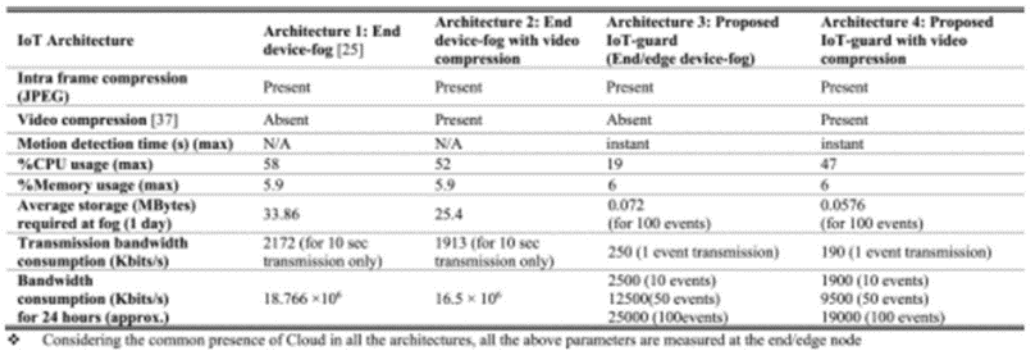


Figure 2.2 Illustration of high-level view of IoT-guard-enabled security management system

The proposed IoT-Guard architecture demonstrates significant resource efficiency compared to traditional surveillance architectures and state-of-the-art frameworks. It achieves substantial savings in CPU usage, memory, storage, bandwidth, and energy consumption by employing an event-driven approach and distributing computational workloads effectively between edge nodes and fog nodes. An IoT video compression algorithm is included to identify if it helps to save a significant amount of transmission energy during transmission of IoT.



IoT Architecture	Architecture 1: End device-fog [25]	Architecture 2: End device-fog with video compression	Architecture 3: Proposed IoT-guard (End/edge device-fog)	Architecture 4: Proposed IoT-guard with video compression
Intra frame compression (JPEG)	Present	Present	Present	Present
Video compression [37]	Absent	Present	Absent	Present
Motion detection time (s) (max)	N/A	N/A	instant	instant
%CPU usage (max)	58	52	19	47
%Memory usage (max)	5.9	5.9	6	6
Average storage (MBytes) required at fog (1 day)	33.86	25.4	0.072 (for 100 events)	0.0576 (for 100 events)
Transmission bandwidth consumption (Kbits/s)	2172 (for 10 sec transmission only)	1913 (for 10 sec transmission only)	250 (1 event transmission)	190 (1 event transmission)
Bandwidth consumption (Kbits/s) for 24 hours (approx.)	18.766×10^6	16.5×10^6	2500 (10 events) 12500(50 events) 25000 (100events)	1900 (10 events) 9500 (50 events) 19000 (100 events)

❖ Considering the common presence of Cloud in all the architectures, all the above parameters are measured at the end/edge node

Figure 2.3 Comparison Among Different IoT-Based Video Surveillance Architectures

Figure 2.3 has shown that with the deployment of the video compression algorithm, the percentage of CPU usage is higher due to the additional computational burden. Whether the IoT-guard architecture is deployed with or without the video compression, the storage requirement is reduced by 99 percent at the fog node. Although the BW efficiency saves only slightly more with video compression, the proposed system can save 42.9 percent more energy if it omits the video compression algorithm.

By leveraging a decentralized edge-fog-cloud approach, IoT-Guard establishes a proactive crime detection and management framework. Crucially, the system optimizes the distribution of computational tasks, directing heavy processing tasks to resource-rich fog nodes while maintaining lightweight operations at edge nodes. This strategic allocation not only ensures real-time functionality but also yields significant cost savings, as well as reductions in energy consumption and bandwidth usage. Additionally, the heightened resource awareness exhibited by edge nodes enables the architecture to accommodate the expansion of surveillance networks efficiently. Overall, IoT-Guard's ability to transmit context-aware multimedia data to fog nodes, coupled with its utilization of low-resource edge nodes, contributes to a substantial decrease in the overall deployment costs of the surveillance system.

Despite the strengths of IoT-Guard, there are a few limitations. The system heavily relies on the interaction between edge and fog nodes. Any issues or failures in this interaction could affect the overall performance and reliability of the system. Furthermore, the system's architecture may lack flexibility in adapting to evolving security requirements or technological advancements. Upgrading or modifying the system to incorporate new features or address emerging threats could be challenging. Lastly, environmental conditions (e.g., weather, lighting) could impact the effectiveness of the surveillance system, potentially leading to false positives or missed detections, especially in outdoor settings or under challenging lighting conditions.

The IoT-Guard framework proposed by Sultana et al. demonstrates substantial resource efficiency and effective task distribution between edge and fog nodes. However, its heavy reliance on edge-fog interaction can affect performance if disrupted, and it is sensitive to environmental conditions. Our solution aims to enhance robustness to node interaction issues and improve adaptability to various environmental conditions to maintain detection accuracy. Additionally, our approach will focus on ensuring seamless operation under different lighting and weather conditions to provide reliable performance in real-world scenarios. For home surveillance, the event-driven approach can be particularly useful in conserving resources while providing timely alerts. Integrating this with our system can enhance real-time detection and reduce unnecessary data processing.

2.3 Abnormality Identification in Video Surveillance System using DCT

Balasundaram et al. [4] introduced a new block-based strategy for abnormality detection in video surveillance systems, focusing on identifying unusual circumstances by analysing pixel-wise frame movement instead of object-based approaches. The proposed strategy utilizes optical flow to extract density and speed of movement and then identifies unusual movement and differences using discrete cosine transform coefficient. The goal is to achieve real-time abnormality detection with a trouble-free block-based Discrete Cosine Transform (DCT) strategy.

The system is evaluated using an airport dataset, and the outcomes of unusual events including various abnormal activities such as sudden dancing, running, and pushing in crowded places are reported. Existing surveillance systems primarily focus on tracking objects in motion, neglecting the actions leading to the movement. Manual intervention is often required for classifying abnormal events, which the proposed system aims to eliminate by automatically recognizing and reporting abnormal events.

The system first identifies changed regions, computes relevant data such as speed and route of movement, and compares them with standardized constraints to detect unusual activities. Hidden Markov Model (HMM) is used for intermediary state representation and comparison with standard action models. Previous techniques for abnormality detection involve segmenting video files, extracting attributes, and applying grouping and resemblance measures, but they struggle with real-time recognition of unusual actions.

The system focuses on analysing action characteristics at the pixel-point level and directing them into pictures that illustrate the action sequence. The data content of these pictures is examined within the "incidence" area by calculating the Discrete Cosine Transform (DCT) coefficients, which are then transmitted to the restricted action area. The system evaluates the performance of objects in every structure by examining and identifying action patterns. It categorizes actions as usual or unusual based on their performance characteristics and compares them to median standard rates over time.

Optical flow, particularly pixel-based optical flow, is utilized to capture action characteristics, although its accuracy can be affected by noise. To overcome this, optical flow approaches use the Lucas-Kanade algorithm. Motion vectors are derived from optical flow, representing the displacement of each pixel in both horizontal and vertical directions. DCT is applied to every segment to compress signal force and calculate DCT coefficients, which are used to compute data structure and entropy. Entropy is contrasted with a threshold rate to determine whether an action is usual or unusual.

Median filtering is employed to handle computation complexity, and abnormal actions are identified when the entropy rate exceeds the threshold rate. A system model of a processed block procedure during an active sequence is depicted, showing the DCT-based abnormal activity detection in surveillance video.

The performance of the proposed approach in pedestrian surveillance videos is evaluated by replicating various unusual circumstances with the help of volunteers, including sudden dancing, running, and pushing in crowded places. The experiment was conducted using videos with a standard screening quality of 720 × 576 pixels at
29 frames per second, ensuring uniform frame rate across the experiment. The system's performance was assessed across different outdoor environmental conditions such as

rain, dim light, and shade. The proposed strategy, which involves sub-dividing each frame into four segments and computing the entropy of Discrete Cosine Transform (DCT) coefficients for each segment, was compared with contemporary works. The median rate of entropy for the first 500 frames was computed, and the threshold for abnormal activities was set thrice higher than this median rate. Any anomalies detected trigger an alert. The abnormal activities identified including human riding a bicycle and driving a truck is shown in Figure 2.4.

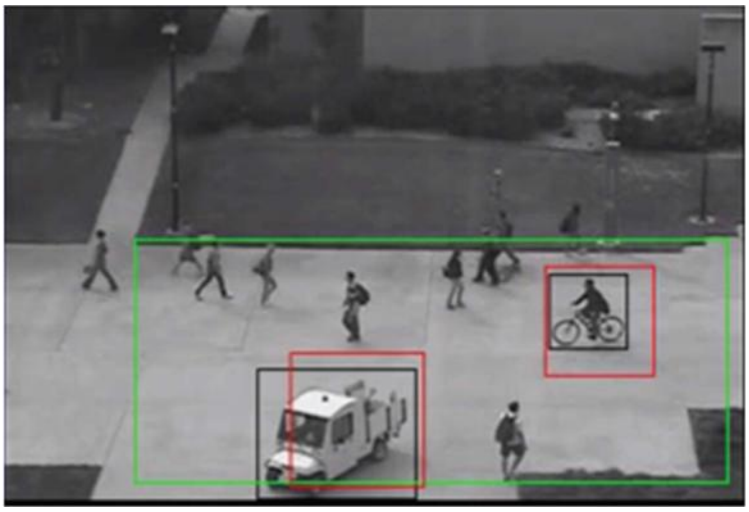
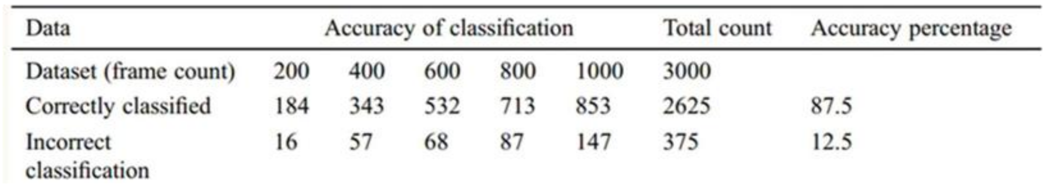


Figure 2.4: A bicycle and truck driven in pedestrian lane detected as anomaly and highlighted in red

The system's performance is evaluated in terms of time complexity for computing and classification accuracy, aiming to enhance classification accuracy rather than speed. No significant latency issues were observed in anomaly detection, and the average time consumed for computing unusual actions was found to be 74.35 milliseconds, inclusive of all intermediate stages. The classification accuracy, a key performance measure, was found to be 87.5%, indicating that anomalies are classified without false alarms.



Data	Accuracy of classification					Total count	Accuracy percentage
Dataset (frame count)	200	400	600	800	1000	3000	
Correctly classified	184	343	532	713	853	2625	87.5
Incorrect classification	16	57	68	87	147	375	12.5

Table 2.1: Classification accuracy-based performance analysis

Moreover, leveraging a block-based approach presents an added advantage, which is the potential for parallel processing during real-time execution. Each block can be autonomously processed without interdependence, thus enhancing the efficiency of anomaly detection in dynamic surveillance environments.

The proposed system exhibits several weaknesses that warrant attention. Firstly, while achieving a classification accuracy of 87.5%, there remains a notable margin for improvement to reduce false alarms and enhance overall reliability. Additionally, the system demonstrates high time complexity, which could hinder real-time performance and scalability, necessitating further optimization efforts to improve computational efficiency. Furthermore, the system's dependency on Discrete Cosine Transform (DCT)-related attributes for anomaly detection may limit its effectiveness in scenarios where such attributes fail to adequately capture or represent anomalous behavior. Furthermore, the scope for future enhancement suggests a need to explore additional attributes beyond DCT, test the system across diverse scenarios, refine accuracy through error modelling, and integrate with more advanced frameworks for smarter surveillance capabilities.

The approach by Balasundaram et al. introduces a novel block-based strategy for real-time abnormality detection using DCT. Despite its strengths in achieving high classification accuracy and enabling parallel processing, the system's high time complexity and dependency on DCT-related attributes limit its applicability. Our proposed solution will focus on reducing computational demands, incorporating additional attributes beyond DCT, and enhancing the system's adaptability to various real-world scenarios. This will ensure a more comprehensive and efficient anomaly detection system. In the context of home surveillance, using optical flow and DCT can help in detecting unusual movements. However, integrating other features such as shape and object recognition will improve the robustness of detecting suspicious activities.

2.4 SlowFast Networks for Video Recognition

Fan et al. [5] introduced SlowFast networks for video recognition, which utilize both Slow and Fast pathways to capture spatial semantics and motion at different temporal resolutions. This approach achieves strong performance in action classification and detection, demonstrating significant improvements over existing methods.

The SlowFast model is designed to address this by incorporating a Slow pathway for semantic information and a Fast pathway for rapidly changing motion. Lateral connections fuse these pathways, allowing each to specialize in spatial or temporal processing. The method draws inspiration from biological studies on retinal ganglion cells and achieves state-of-the-art results on major video recognition benchmarks.

Through comprehensive ablation experiments, the effectiveness of SlowFast networks is demonstrated, surpassing previous systems in the literature.

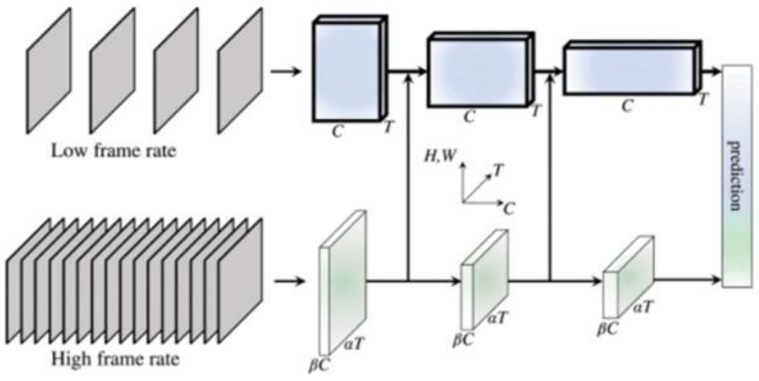


Figure 2.5 A SlowFast network

The SlowFast Networks architecture is characterized by two distinct pathways operating at different frame rates. The Slow pathway processes video clips sparsely, with a large temporal stride, capturing semantic information evolving slowly over time. In contrast, the Fast pathway operates at a higher frame rate, sampling frames densely to capture rapidly changing motion. The Fast pathway maintains high temporal resolution features throughout the network hierarchy and uses significantly lower channel capacity compared to the Slow pathway, making it lightweight. Lateral connections fuse information from both pathways, ensuring that each pathway is aware of the representations learned by the other. The architecture allows for flexible instantiation with different backbones and implementation specifics.

The system demonstrates impressive performance across various action classification and detection datasets like Kinetics-400, Kinetics-600, and Charades, particularly without relying on ImageNet pre-training. Despite this high accuracy, SlowFast networks maintain computational efficiency, owing to innovative training techniques and the integration of both Slow and Fast pathways. This design choice not only enhances efficiency but also enables the networks to capture both spatial and temporal features effectively, leading to improved performance compared to architectures relying solely on one pathway. It uses synchronized SGD training and employs lightweight pathways, such as the Fast pathway, which contributes to efficient processing.

Moreover, SlowFast networks demonstrate flexibility in handling different input sampling rates and backbone architectures, allowing for adaptation to various datasets and tasks. This adaptability is particularly advantageous in real-world applications where datasets may vary in terms of video length, frame rate, and content complexity. By adjusting the input sampling rates and backbone architectures, SlowFast networks can effectively capture temporal dependencies and spatial features across diverse datasets, thus improving their robustness and generalization capabilities.

However, SlowFast networks also possess several weaknesses, including complexity in design choices and hyperparameter tuning, which may pose challenges in understanding and implementing them effectively. Additionally, achieving optimal performance with SlowFast networks may require extensive training procedures and parameter tuning, potentially limiting their generalization to other domains or tasks.

Furthermore, SlowFast networks may be sensitive to hyperparameter choices, necessitating careful experimentation to fully exploit their capabilities. Thus, while SlowFast networks represent a significant advancement in video recognition, their strengths and weaknesses highlight the need for careful consideration and experimentation to maximize their potential.

The SlowFast networks introduced by Fan et al. showcase significant improvements in video recognition through innovative dual-pathway design. However, the complexity of the model and the need for extensive hyperparameter tuning can limit its accessibility and generalization. Our solution aims to simplify design choices and reduce training complexity while maintaining high accuracy and efficiency, ensuring broader applicability in various video recognition tasks. Applying SlowFast networks to home surveillance can enhance the detection of rapid and subtle movements by processing at different temporal resolutions. This dual-pathway design can be particularly useful in identifying quick intrusions and slower suspicious behaviors, improving overall detection accuracy.

2.5 Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition

Zhang et al. [6] introduced a novel approach, the two-stream adaptive graph convolutional network (2s-AGCN), for skeleton-based action recognition. It addresses limitations in existing methods by proposing a data-driven method to learn the graph structure adaptively, improving the model's flexibility and generality across diverse datasets. Additionally, the incorporation of second-order information, capturing bone lengths and directions, enhances discriminative features for action recognition.

The two-stream framework integrates both first-order and second-order information, further boosting performance. Extensive experiments on NTU-RGBD and Kinetics-Skeleton datasets demonstrate the superiority of the proposed model, achieving state-of-the-art results. Overall, the contributions include adaptive graph construction, utilization of second-order information, and significant performance improvements on large-scale action recognition datasets.

In the graph construction phase, the raw skeleton data, represented as joint- coordinate vectors, are structured into a spatiotemporal graph, capturing spatial and temporal relationships among joints. The graph consists of vertices representing joints and edges representing natural connections between them. Temporal edges connect corresponding joints across consecutive frames. Subsequently, multiple layers of spatiotemporal graph convolution operations are applied to extract high-level features, followed by a global average pooling layer and softmax classifier for action prediction. The graph convolution operation involves sampling areas for convolution, where a weighting function provides weight vectors based on input, and a mapping function ensures unique weight vectors for each vertex.

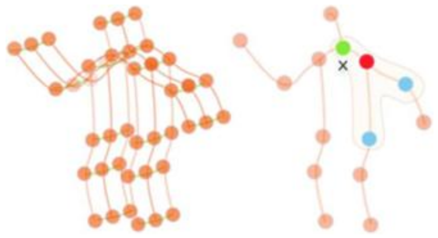


Figure 2.6 Illustration of the spatiotemporal graph used in STGCN and the mapping strategy.

Implementation-wise, the feature map is a tensor, and convolution operations are performed to extract connected vertexes and their importance. For the temporal dimension, convolution is straightforward due to the fixed number of neighbors for each vertex. Overall, the approach leverages GCNs to effectively model spatial and temporal dependencies in skeleton data for accurate action recognition.

Two significant action recognition datasets, the NTU-RGBD and Kinetics-Skeleton are used to evaluate the proposed two-stream adaptive graph convolutional network (2s-AGCN). The experiments were conducted using the PyTorch deep learning framework, employing stochastic gradient descent (SGD) with Nesterov momentum

(0.9) as the optimization strategy. A batch size of 64 was used, with cross-entropy serving as the loss function for backpropagation of gradients. The weight decay was set to 0.0001 to prevent overfitting.

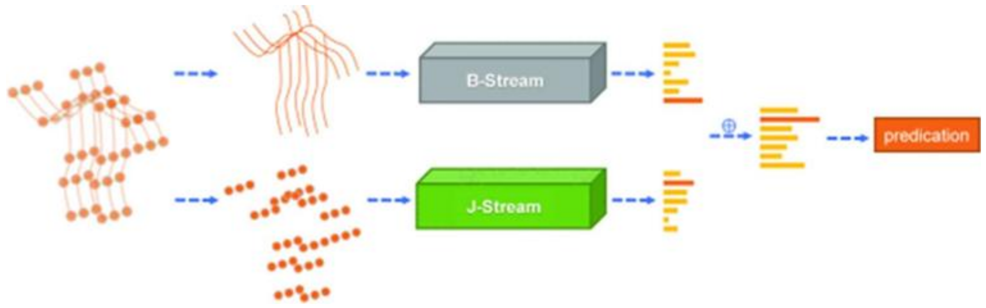


Figure 2.7 Illustration of the overall architecture of the 2s-AGCN.

For the NTU-RGBD dataset, samples were ensured to have at most two people, with padding applied if necessary. The maximum number of frames per sample was set to 300, with shorter samples repeated to meet this length. The learning rate was initially set to 0.1 and reduced by a factor of 10 at the 30th and 40th epochs, with training concluding at the 50th epoch.

Similarly, for the Kinetics-Skeleton dataset, input tensors consisted of 150 frames with 2 bodies each, following the data augmentation methods outlined in prior work. The learning rate schedule mirrored that of the NTU-RGBD dataset, with reductions at the 45th and 55th epochs and training concluding at the 65th epoch.

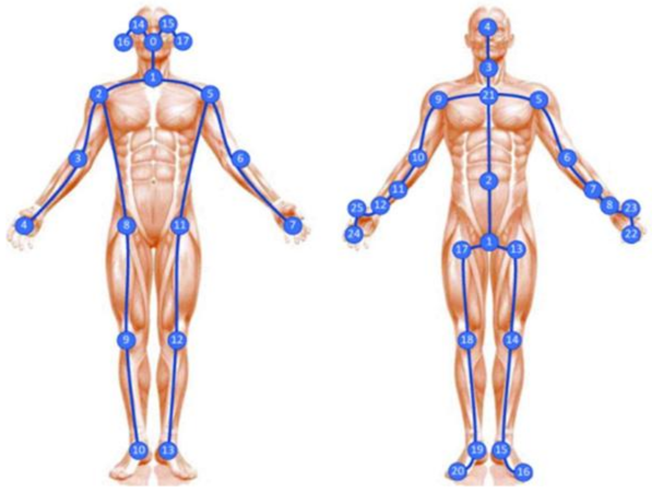
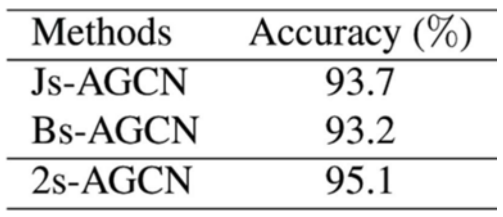


Figure 2.8 Joint label of the Kinetics-Skeleton dataset the NTU-RGBD dataset.

The proposed two-stream adaptive graph convolutional network (2s-AGCN) demonstrates notable strengths in skeleton-based action recognition, as evidenced by comprehensive experiments conducted on the NTU-RGBD and Kinetics-Skeleton datasets. Firstly, the model's architecture, illustrated in Figure 2.10, incorporates two streams whose scores are combined for final prediction, enabling a robust approach to recognizing actions. Moreover, the adaptability of the graph convolutional block proves beneficial for action recognition, with the model achieving its best performance when all three types of graphs are utilized. The visualization of learned graphs further underscores the flexibility and data-driven nature of the model, showcasing its ability to capture nuanced relationships between joints in the skeleton data. Additionally, the incorporation of second-order information through a two- stream framework significantly enhances performance, surpassing one-stream-based methods, as demonstrated in Table 2.2.



Methods	Accuracy (%)
Js-AGCN	93.7
Bs-AGCN	93.2
2s-AGCN	95.1

Table 2.2 Comparisons of the validation accuracy with different input modalities.

Furthermore, the model exhibits state-of-the-art performance when compared with other skeleton-based action recognition methods. This indicates its superiority in accurately recognizing actions across different datasets, NTU-RGBD and Kinetics-Skeleton. The robustness of the proposed system is further validated by its ability to generalize well across datasets and surpass existing approaches by a considerable margin.

However, despite these strengths, the system may have some weaknesses. Firstly, while the model achieves state-of-the-art performance, it is essential to consider computational complexity and resource requirements, especially given the intricate architecture and the need for extensive training. Additionally, the effectiveness of the proposed system heavily relies on the quality and diversity of the training data, raising concerns about its performance in real-world scenarios where data may be limited or unrepresentative. Moreover, while the model's adaptability to different datasets is highlighted, its performance in highly dynamic or challenging action recognition scenarios remains to be fully explored. Finally, the interpretability of the learned graph structures and their implications for real-world action recognition tasks may require further investigation for practical deployment.

In conclusion, the proposed 2s-AGCN system demonstrates significant advancements in skeleton-based action recognition, leveraging adaptive graph convolutional networks and a two-stream framework to achieve state-of-the-art performance across diverse datasets. While its strengths lie in its flexibility, robustness, and superior performance, potential weaknesses include computational complexity, data dependency, and challenges in real-world deployment and interpretability. Addressing these concerns could further enhance the system's applicability and efficacy in practical action recognition scenarios.

The two-stream adaptive graph convolutional network (2s-AGCN) by Zhang et al. demonstrates notable strengths in skeleton-based action recognition, with superior performance on large-scale datasets. However, the model's complexity and reliance on high-quality training data present challenges. Our approach will focus on reducing computational demands, improving data efficiency, and enhancing real-world applicability. For home surveillance, 2s-AGCN can be adapted to recognize human activities and detect unusual behaviors by leveraging skeleton data. This can be particularly useful in distinguishing between normal and suspicious movements, enhancing the system's ability to detect potential threats accurately.

2.6 NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding

Liu et al. [7] introduced a large-scale dataset for RGB+D human action recognition comprising over 114,000 RGB+D video samples captured from 106 human subjects across various age groups and cultural backgrounds. The dataset includes RGB videos, depth sequences, skeleton data, and infrared frames captured from 155 different camera viewpoints under diverse environmental conditions. This diversity enables more realistic evaluations of 3D-based action analysis methods and facilitates the application of data-driven learning techniques like deep learning.

The introduction of depth sensors like Microsoft Kinect, Intel RealSense, and Asus Xtion has propelled computer vision into the realm of 3D vision, enabling advancements in 3D object recognition, scene understanding, and activity analysis. However, 3D action recognition faces challenges due to the lack of large-scale benchmark datasets with diverse samples representing various action classes in real-world scenarios.

Existing 3D action recognition benchmarks suffer from limitations such as a small number of subjects, a narrow range of performers' ages, limited action categories, and restricted camera views. These constraints hinder the development and evaluation of robust and generalizable 3D action recognition models, particularly deep learning approaches.

State-of-the-art 3D action recognition approaches are evaluated on the NTU RGB+D 120 dataset, demonstrating the effectiveness of deep models in activity analysis. Fusion techniques across different data modalities, such as RGB, depth, and skeleton data, are also evaluated to leverage their complementary information for more accurate action recognition.

Furthermore, the paper investigates a novel one-shot 3D action recognition problem using the proposed dataset. An Action-Part Semantic Relevance-aware (APSR) framework is introduced to address this task by leveraging semantic relevance between body parts and action classes. By emphasizing the relevant body parts based on semantic guidance, the APSR framework improves the performance of one-shot action recognition.

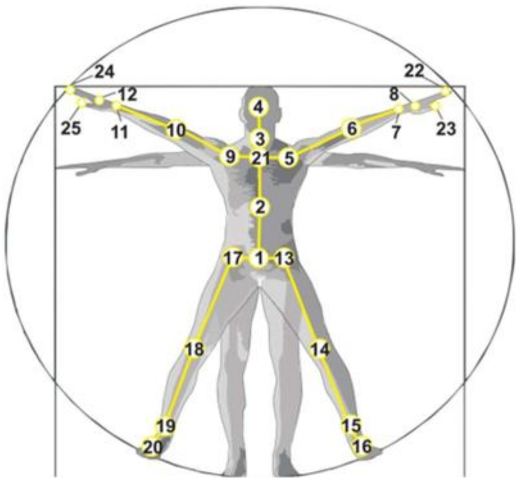
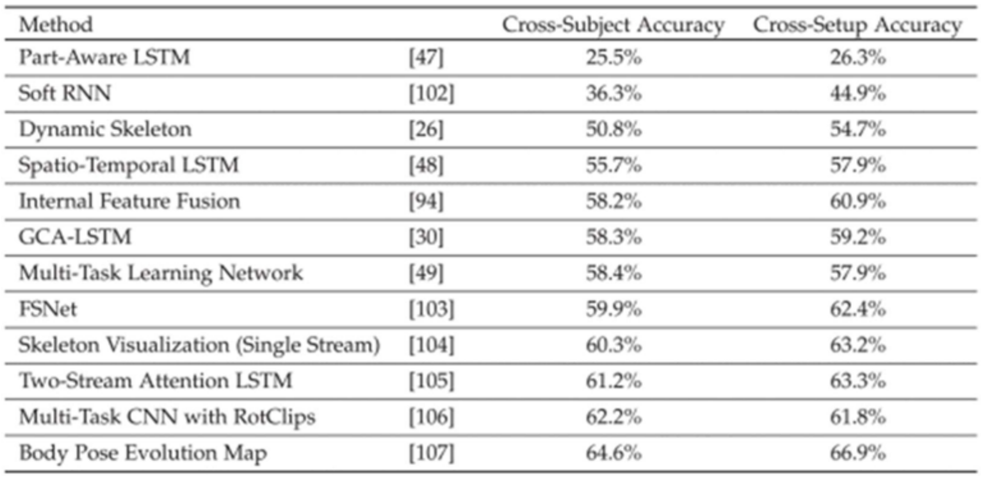


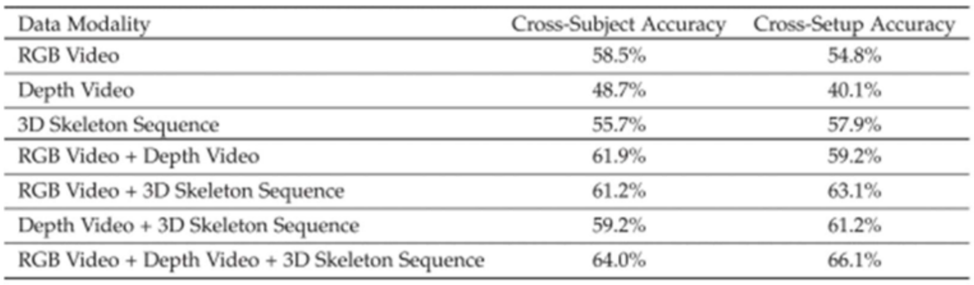
Figure 2.9 Illustration of 25 body joints in dataset

The proposed dataset’s strengths include its comprehensive evaluation of state-of- the-art methods, encompassing diverse approaches such as Part-Aware LSTM, Soft RNN, and Multi-Task CNN with RotClips. By comparing different methods using cross-subject and cross-setup criteria, the system provides a nuanced understanding of their performance under various conditions. Additionally, the evaluation of different data modalities (RGB, depth, and 3D skeleton data) offers insights into their respective strengths and weaknesses, highlighting the importance of modality fusion for improved performance. The system also demonstrates the effectiveness of using a large training set, showing clear improvements in action recognition accuracy with increased training data size.



Method		Cross-Subject Accuracy	Cross-Setup Accuracy
Part-Aware LSTM	[47]	25.5%	26.3%
Soft RNN	[102]	36.3%	44.9%
Dynamic Skeleton	[26]	50.8%	54.7%
Spatio-Temporal LSTM	[48]	55.7%	57.9%
Internal Feature Fusion	[94]	58.2%	60.9%
GCA-LSTM	[30]	58.3%	59.2%
Multi-Task Learning Network	[49]	58.4%	57.9%
FSNet	[103]	59.9%	62.4%
Skeleton Visualization (Single Stream)	[104]	60.3%	63.2%
Two-Stream Attention LSTM	[105]	61.2%	63.3%
Multi-Task CNN with RotClips	[106]	62.2%	61.8%
Body Pose Evolution Map	[107]	64.6%	66.9%

Table 2.3 Evaluation of different methods.



Data Modality	Cross-Subject Accuracy	Cross-Setup Accuracy
RGB Video	58.5%	54.8%
Depth Video	48.7%	40.1%
3D Skeleton Sequence	55.7%	57.9%
RGB Video + Depth Video	61.9%	59.2%
RGB Video + 3D Skeleton Sequence	61.2%	63.1%
Depth Video + 3D Skeleton Sequence	59.2%	61.2%
RGB Video + Depth Video + 3D Skeleton Sequence	64.0%	66.1%

Table 2.4 Evaluation of different data modalities.

Furthermore, the detailed analysis according to data modalities and methods provides valuable insights into the challenges and limitations of each approach, informing future research directions. For instance, the analysis reveals the effectiveness of skeleton data in capturing view-invariant features and the challenges posed by object-related actions. Moreover, the proposed APSR framework for one- shot recognition showcases the system's adaptability to novel action classes, indicating its potential for real-world applications. Overall, the system's strengths lie in its comprehensive evaluation framework, detailed analysis, and potential for addressing real-world challenges in 3D action recognition.

However, the system also exhibits certain weaknesses. One limitation is the reliance on hand-crafted features for some methods, which may limit their ability to generalize across different datasets or capture complex patterns in the data. Additionally, while the system evaluates performance across different data modalities, it may not fully explore the potential of emerging modalities such as infrared sequences. The analysis of action-wise performance highlights challenges in accurately recognizing actions involving fine-grained hand gestures or interactions with objects, indicating areas where further improvement is needed.

Moreover, the system's focus on one-shot recognition, while promising, may overlook the broader context of continuous action recognition and the challenges associated with temporal modelling. Lastly, while the dataset size is a strength, it also presents challenges in terms of computational resources required for training and evaluating models, which may limit accessibility for researchers with limited resources. In short, while the system provides valuable insights and contributions to the field of 3D action recognition, addressing these weaknesses could further enhance its impact and applicability.

The dataset and evaluation framework introduced by Liu et al. provide comprehensive insights into 3D action recognition. However, the size and diversity of the dataset present computational challenges. Our solution will optimize computational efficiency and leverage automated feature extraction techniques to address these challenges. For home surveillance, using a large-scale, diverse dataset for training can significantly improve the system's ability to recognize various activities and adapt to new types of threats. By ensuring that the training data includes a wide range of home intrusion scenarios, the system can achieve high accuracy and reliability in real-world applications.

Chapter 3
System Methodology/Approach

3.1 System Design Diagram/Equation

3.1.1 System Architecture Diagram

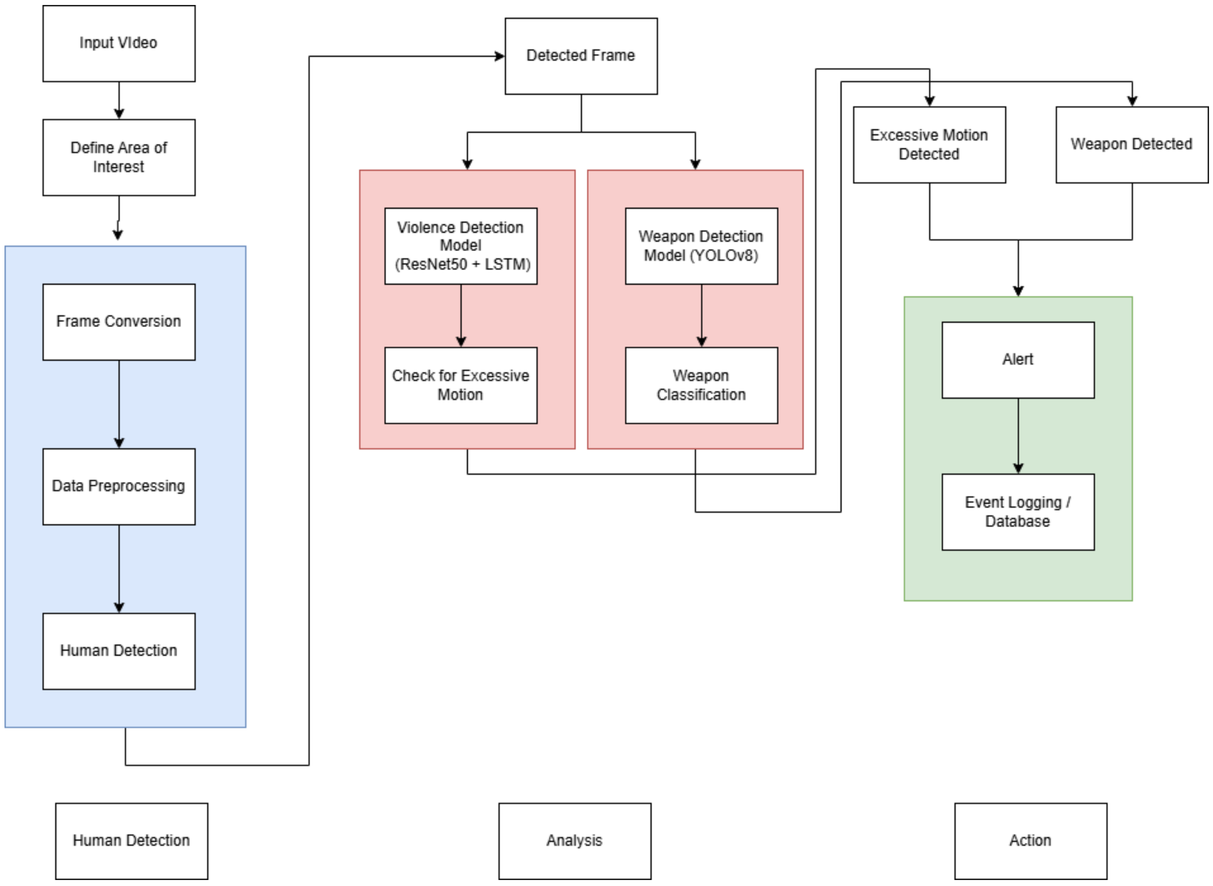


Figure 3.1.1 Flow of system methodology

The realization of the proposed intelligent home surveillance system follows a structured development methodology integrating real-time human detection, violence analysis, and weapon detection. The system is designed to provide a robust and automated security solution that minimizes false alarms while enhancing situational awareness through deep learning and threaded processing techniques.

The methodology is organized into three main phases: detection, analysis, and response. The process begins with the user defining a Region of Interest (ROI) within the camera feed—critical for focusing the surveillance on specific zones and improving the relevance of subsequent detections. Live video input is captured and converted into individual frames, which then undergo preprocessing such as resizing and normalization to optimize them for model inference.

Following preprocessing, each frame is passed through a YOLOv8-based human detection module. Only frames containing detected persons within the defined ROI are considered for further analysis. At this point, two parallel detection pipelines are activated via separate processing threads to ensure real-time performance with low latency.

In the first pipeline, a sliding window of recent frames is analyzed using dense optical flow and a pre-trained ResNet50 + LSTM model to detect violent or abnormal motion patterns. In the second pipeline, the system performs inference using a pre-trained YOLOv8 model to detect weapons such as knives, guns, or similar threatening objects.

If either excessive motion or a weapon is detected, the system transitions into the response phase. A real-time on-screen alert is generated, displaying the event type (e.g., "Violence" or "Weapon"), confidence level, and timestamp. These events are simultaneously logged for future review, enabling evidence preservation and pattern analysis.

This approach ensures that the system not only automates home surveillance but does so with precision and efficiency. The focus remains on software-level threat detection and alerting; no integration with physical countermeasures or third-party hardware is included in this scope.

3.1.2 Use Case Diagram and Description

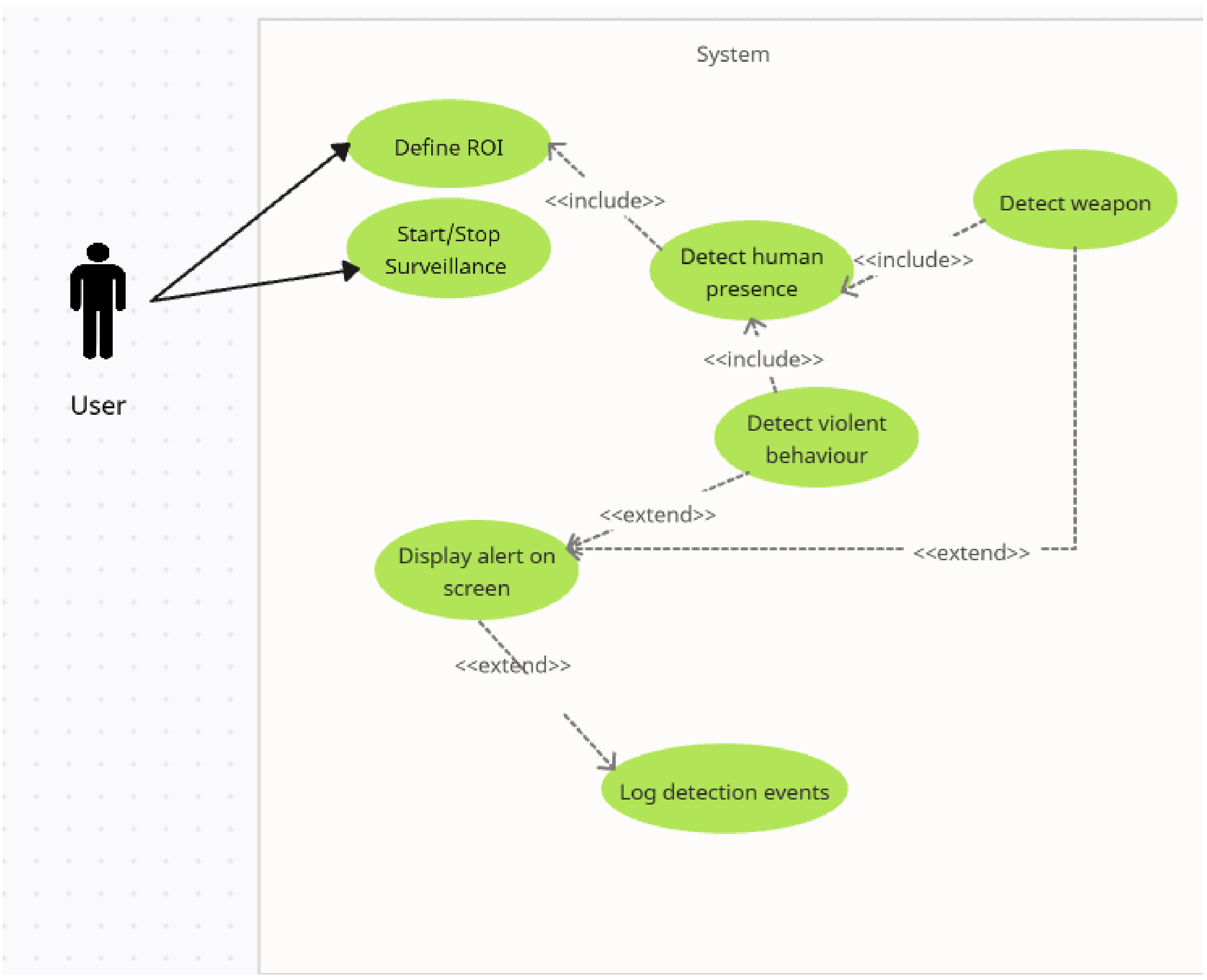


Figure 3.1.2 Use case diagram

The use case diagram for the proposed surveillance system illustrates the interactions between the primary actor—the User (Homeowner or Security Personnel)—and the System. The user is responsible for initializing and controlling the surveillance operations, while the system performs all real-time monitoring and intelligent detection tasks.

The use case begins with the user defining the Region of Interest (ROI), which specifies the area to be monitored for human presence. This is a prerequisite step and is included in the Detect Human Presence use case using a <<include>> relationship. This inclusion reflects that human detection cannot be performed unless the ROI has already been specified. Once the ROI is set, the user can initiate or terminate the surveillance through the Start/Stop Live Surveillance use case.

During active surveillance, the system continuously monitors the defined ROI for the presence of humans. If a person is detected, the system proceeds to perform two concurrent actions. First, it conducts motion analysis across a sequence of frames to Detect Violent Behavior, using a ResNet50 + LSTM model. Second, it performs real-time object classification using a YOLOv8 model to Detect Weapons such as knives or pistols. Both detections use cases extend from Detect Human Presence via <<extend>> relationships, as they are conditional actions that occur only if a person is detected.

If either violence or weapon presence is confirmed, the system triggers the Display Alert on Screen use case to notify the user. Additionally, the system will execute the Log Detection Events use case to store detailed records of the incident in a database for future reference and review.

The use case diagram thus effectively captures the flow of control, the dependencies between actions, and the system’s reactive behavior upon detecting potential threats, ensuring a clear and comprehensive model of user-system interaction in the surveillance process.

3.1.3 Activity Diagram

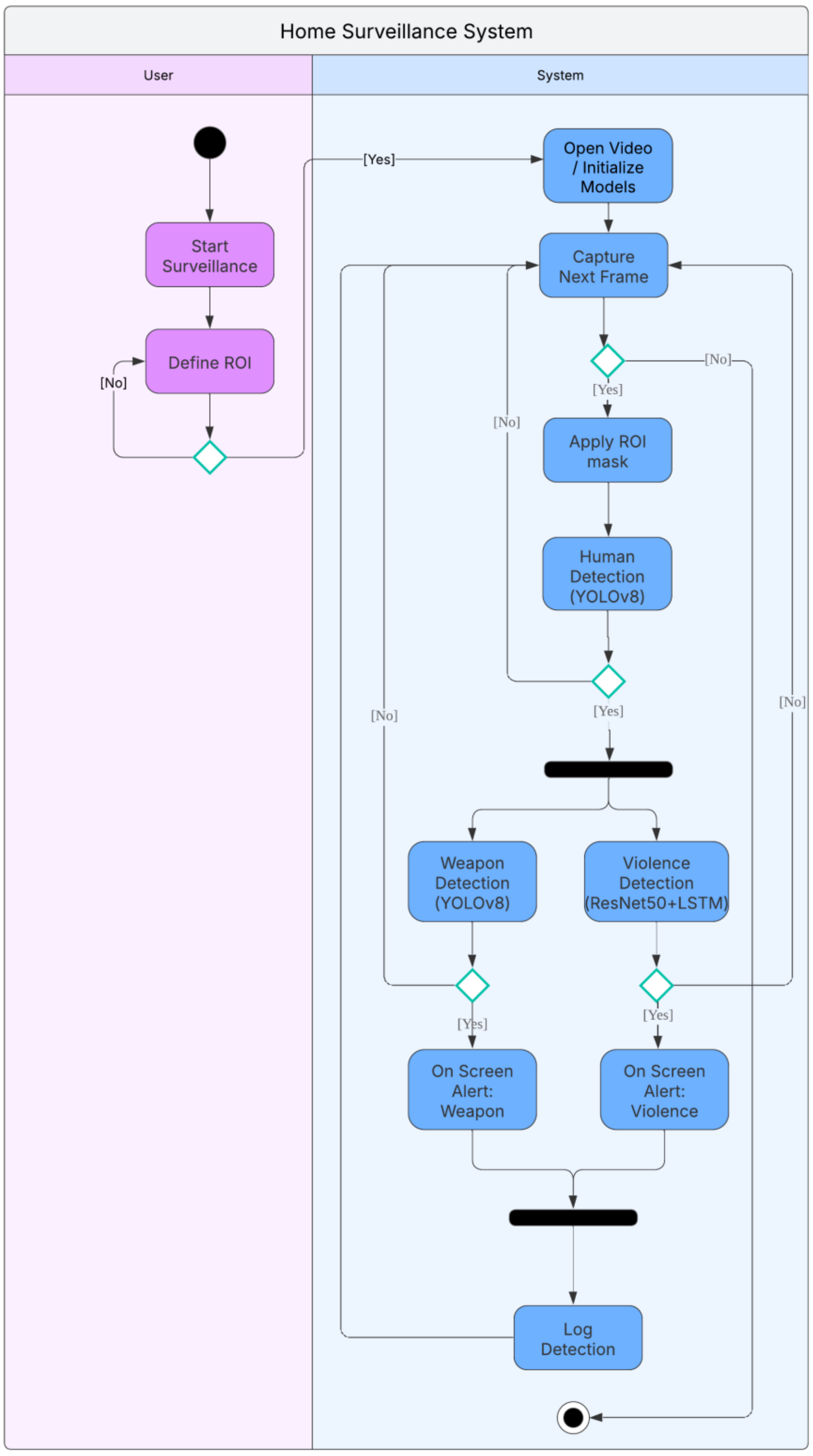


Figure 3.1.3 Activity diagram

The workflow of the intelligent surveillance system is depicted in Figure 3.1.3 as an activity diagram, illustrating the complete process from system initialization to real-time threat detection and response. The process begins when the user initiates the system, prompting an initialization phase where the user is required to define a Region of Interest (ROI). This ROI constrains human detection to specific areas within the video feed, ensuring that analysis is limited to zones of concern. The system loops this step until the ROI is properly defined, emphasizing user control and precision.

Following successful ROI setup, the system enters its core operational loop, continuously capturing video frames. For each frame, the system first verifies whether additional frames are available. If so, it proceeds with analysis by performing human detection using a YOLOv8 model. If no individuals are detected within the ROI, the frame is skipped, and the system advances to the next one, conserving computational resources.

When a person is detected, the system launches two parallel detection processes. The first involves violence detection, which leverages a ResNet50 combined with an LSTM model to analyze temporal motion patterns across a sequence of frames. The second is weapon detection, which employs another YOLOv8 model to identify potentially dangerous objects such as guns or knives. Each detection path includes a confidence-based decision mechanism: if violent behavior is identified with a confidence score exceeding the predefined threshold, the system triggers an on-screen alert and logs the event. Similarly, if a weapon is detected with sufficient confidence, a weapon alert is displayed and recorded.

If no threat is identified in either branch, the system skips alert generation but continues monitoring by capturing the next frame. This loop enables real-time processing and threat recognition with continuous feedback. The system remains active until no further frames are available, at which point the session concludes. Overall, the activity diagram captures the decision-driven, real-time operation of the surveillance system, emphasizing threat prioritization, user-defined parameters, and seamless integration between detection and response.

Chapter 4

System Design

4.1 System Block Diagram

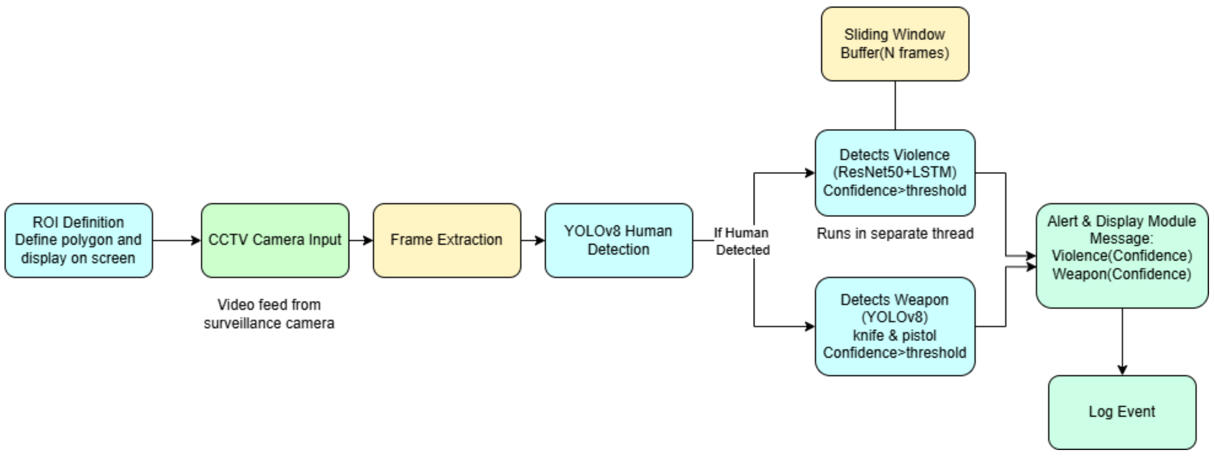

Sliding Window Buffer(N frames)
Detects Violence (ResNet50+LSTM) Confidence>threshold
Runs in separate thread
ROI Definition Define polygon and display on screen
CCTV Camera Input
Frame Extraction
YOLOv8 Human Detection
If Human Detected
Video feed from surveillance camera
Alert & Display Module Message: Violence(Confidence) Weapon(Confidence)
Detects Weapon (YOLOv8) knife & pistol Confidence>threshold
Log Event

Figure 4.1 System block diagram

Figure 4.1 illustrates the system architecture of the proposed Home Surveillance System with Violence and Weapon Detection. To ensure that only the relevant area of each video is analysed, the pipeline first invokes the ROI Definition Module, in which the user interactively draws a polygon on the first frame of the selected video. The user must specify at least three points; if fewer than three points are provided, the ROI is considered invalid, and the system will prompt for redefinition before proceeding. Once a valid ROI is established, its coordinates remain fixed for the remainder of that session.

The system then proceeds to the Video Dataset Input stage, where pre-recorded CCTV footage (MP4 files at 30 fps) is loaded. The Frame Extraction module samples frames at a fixed interval (e.g., every 5th frame), resizes them to 224×224 pixels, and normalizes pixel values for downstream processing.

Each extracted frame is passed through the YOLOv8 Person Detection module, which applies the ROI mask and locates any human figures. Frames without any detected person within the ROI are discarded to conserve computational resources.

When a person is detected, two parallel detection pipelines are launched:

1. Violence Detection Module

A Sliding Window Buffer maintains the most recent 16 person-positive frames. This clip is fed into a ResNet50 + LSTM network that captures both spatial and temporal cues. If the model's probability exceeds the configured confidence threshold, the system outputs violence (confidence).

2. Weapon Detection Module

The same ROI-masked frame is also sent to a second YOLOv8 model, fine-tuned to recognize handled weapons—specifically pistols and knives. Detections with confidence above the threshold produce weapon (confidence) outputs.

Finally, the Alert & Display Module overlays all confirmed detections—including person bounding boxes, violence (conf), and weapon (conf)—onto the live feed. Critical alerts are highlighted in real time, and each event triggers the Event Logging Module, which records the timestamp, event type, confidence score for audit, reporting, or legal purposes. This end-to-end, multi-threaded design ensures efficient resource use, accurate threat detection, and comprehensive traceability.

4.2 System Components Specifications

This section provides a detailed specification for each component in the system block diagram. For every module, we outline its purpose, inputs, outputs, key parameters, implementation libraries, and performance characteristics—enabling a clear blueprint for reproduction.

4.2.1 ROI Definition Module

Function:

Allow the user to draw a polygon on the first video frame to define the Region of Interest (ROI) before any detection runs.

Implementation:

Uses OpenCV mouse callbacks to record N clicks (N≥3). Then, connects them into a polygon. Blocks further processing until at least 3 points are defined.

4.2.2 Surveillance Camera Video Input

Function:

Supply pre-recorded surveillance footage for testing and validation of the detection pipeline.

Source:

UCF-Crime dataset, specifically videos labelled as “fighting” and “shooting”

Format & Specs:

Container: MP4 (H.264 video codec)

Frame Rate: 30 fps

4.2.3 Frame Extraction

Function:

Decode the video stream and sample frames at a fixed interval for downstream analysis.

Library: OpenCV

Key Parameters:

FRAME_SKIP = 5: only every 5th frame is retained

Resize: frames are resized to IMG_SIZE = (224, 224) to match model input

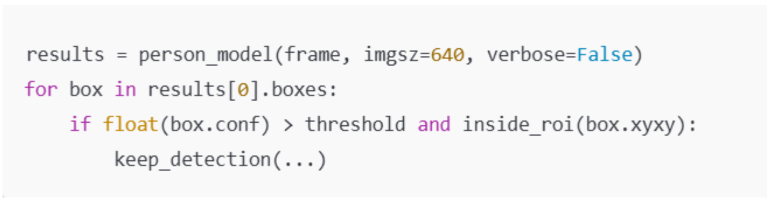
Normalization: pixel values divided by 255.0

4.2.4 YOLOv8 Person Detection

Function: Identify human figures in each extracted frame and filter by a user-defined Region of Interest (ROI).

Model: Ultralytics YOLOv8, fine-tuned for the “person” class.

Implementation:



```
results = person_model(frame, imgsz=640, verbose=False)
for box in results[0].boxes:
    if float(box.conf) > threshold and inside_roi(box.xyxy):
        keep_detection(...)
```

Figure 4.2.1 Person detection module

4.2.5 Sliding Window Buffer

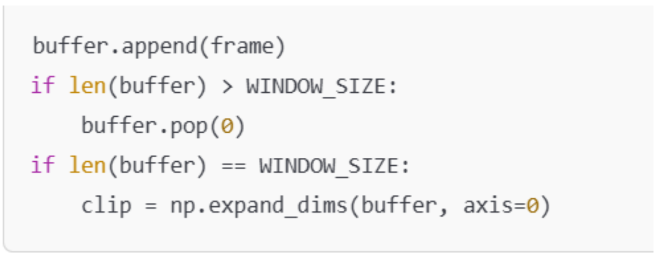
Function:

Maintain a rolling buffer of the most recent person-positive frames for temporal analysis.

Structure:

FIFO queue of length WINDOW_SIZE = 16.

Implementation:



```
buffer.append(frame)
if len(buffer) > WINDOW_SIZE:
    buffer.pop(0)
if len(buffer) == WINDOW_SIZE:
    clip = np.expand_dims(buffer, axis=0)
```

Figure 4.2.2 Sliding window buffer module

4.2.6 ResNet50 + LSTM Violence Detection

Function:

Classify a sequence of frames as violent or non-violent by modelling both spatial and temporal features.

Architecture Details:

1. Spatial feature extractor: TimeDistributed(ResNet50)
    - Pretrained on ImageNet; first 100 layers frozen.
2. Pooling: TimeDistributed(GlobalAveragePooling2D()) → sequence of feature vectors.
3. Temporal model: LSTM
4. BatchNormalization() after LSTM

4.2.7 YOLOv8 Weapon Detection

Function:

Detect and classify handled weapons—specifically pistols and knives—in frames containing a person.

Model:

Ultralytics YOLOv8 fine-tuned on a custom dataset of "pistol" and "knife" classes.

Aggregation Strategy:

Process all sampled frames; track the highest confidence seen across the video and return a single clip-level weapon score.

4.2.8 Alert & Display Module

Function:

Visually annotate the video feed with detection results and raise alerts.

Outputs: Annotated frames for labels such as violence (confidence score) or weapon (confidence score) displayed in an OpenCV window.

4.2.9 Event Logging Module

Function:

Record every detection event—person, violence, and weapon—with precise contextual details for later review, auditing, or evidence.

Implementation:

Uses Python's built-in logging module, configured at the top of the script:

```
logging.basicConfig(
    filename='detection_events.log',
    level=logging.INFO,
    format='%(asctime)s | %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)
```

Figure 4.2.3 Event Logging Module

All log entries are appended to the file detection_events.log.

Log Record Format:

```
YYYY-MM-DD HH:MM:SS | <EventType> | Frame: <frame_idx> | Conf: <confidence> | Box: <box>
```

Figure 4.2.4 Log Record Format

EventType: One of "Person", "Weapon", or "Violence".

Frame: The index of the video frame when the event was detected.

Conf: The model's confidence score, formatted to two decimal places.

Box: For "Person" and "Weapon", the bounding-box coordinates (x1, y1, x2, y2); for "Violence", "N/A" since it applies to a multi-frame clip.

4.3 Circuit and Components Design

This section outlines the design and configuration of each core component in the home surveillance system, with a focus on the computational architecture and how each part contributes to the system's functionality. It also includes the methodology used for training the models responsible for violence and weapon detection.

1. Surveillance Video Input

The system supports both real-time video feed from a connected webcam and offline testing using pre-recorded surveillance footage. Video capture is handled through the OpenCV library, which streams frames into the detection pipeline.

For evaluation purposes, the UCF-Crime dataset is employed. It is a large-scale, real-world surveillance video dataset comprising 13 categories of anomalies (e.g., fighting, robbery, abuse) and normal activities. The dataset provides unconstrained camera angles, low lighting conditions, and varied real-world environments, making it suitable for testing the generalizability of the trained models. In this project, only the fighting and shooting videos from UCF-Crime were used for frame-level testing of the violence and weapon detection modules.

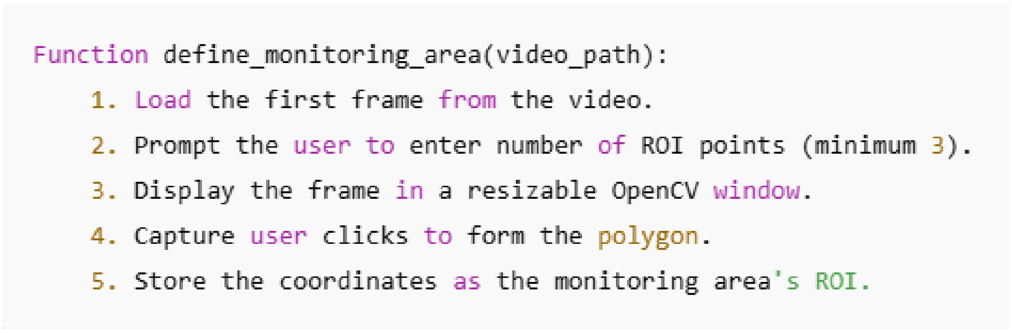
2. Region of Interest (ROI) module

Before the system begins operation, users are required to define a Region of Interest (ROI) that specifies the active monitoring zone within the video feed. This ROI acts as a spatial filter, focusing computational resources on the area most relevant to potential intrusion or abnormal behaviour.

The ROI is created by manually drawing a polygon on the first frame of the video stream. Users must select at least three points to form a valid polygon. If fewer than three points are selected, the system will prompt the user to redefine the region until a valid polygon is submitted. This interactive configuration step ensures flexibility, allowing the surveillance system to be tailored to various camera views and environments (e.g., entrances, hallways, parking lots).

Once defined, the ROI polygon is overlaid on the video stream using OpenCV. During live detection, only objects (such as people or weapons) whose bounding box centroids fall within this ROI are passed to the violence and weapon detection modules for further analysis. Objects outside the ROI are ignored to conserve computational resources and prevent irrelevant areas—like walls, open spaces, or background movement—from triggering false detections.

By focusing analysis strictly within the ROI, this module significantly improves both the efficiency and accuracy of the system. It reduces unnecessary inference calls, lowers latency in real-time processing, and minimizes false positives caused by background noise or movement in non-critical areas.



```
Function define_monitoring_area(video_path):
    1. Load the first frame from the video.
    2. Prompt the user to enter number of ROI points (minimum 3).
    3. Display the frame in a resizable OpenCV window.
    4. Capture user clicks to form the polygon.
    5. Store the coordinates as the monitoring area's ROI.
```

Figure 4.2.5 ROI module pseudocode

3. Person Detection module

This module forms a critical component of the surveillance system, responsible for identifying the presence of humans in each frame of the video feed. It uses the YOLOv8 (You Only Look Once version 8) object detection model, a state-of-the-art deep learning architecture known for its balance of speed and accuracy in real-time detection scenarios.

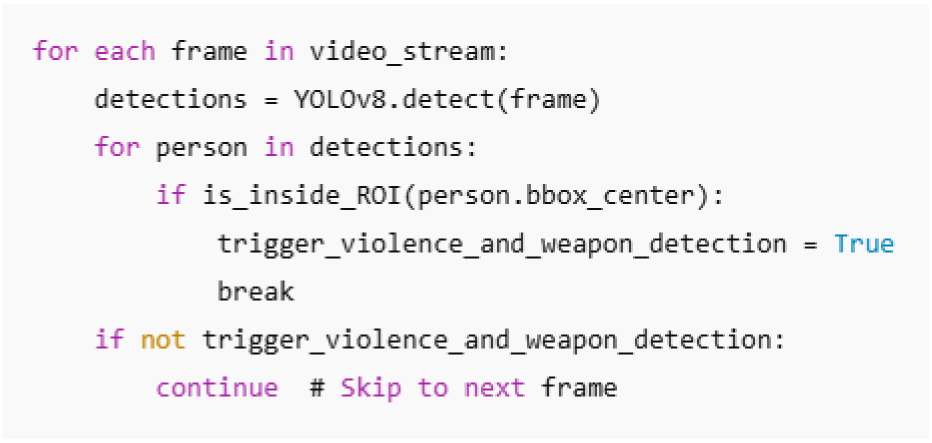
The YOLOv8 model used here is pre-trained on the COCO dataset, a large-scale dataset containing over 80 object categories, including the “person” class. For each input frame, YOLOv8 generates the bounding boxes indicating the location of detected persons.

To improve both performance and computational efficiency, the detection is restricted to a user-defined Region of Interest (ROI). Only humans detected within the ROI polygon are considered valid.

Person detection acts as a conditional gatekeeper for downstream processing, if no persons are detected inside the ROI, the system skips both the violence and weapon detection stages for that frame. This reduces unnecessary computation and helps prevent false positives triggered by irrelevant background motion or objects. If one or more persons are detected, the corresponding frame is passed to both:

- The Violence Detection Module (ResNet50 + LSTM)
- The Weapon Detection Module (YOLOv8 custom model)

This selective activation ensures that the system remains responsive and resource-efficient, especially when deployed in real-time environments with limited computational power.



```
for each frame in video_stream:
    detections = YOLOv8.detect(frame)
    for person in detections:
        if is_inside_ROI(person.bbox_center):
            trigger_violence_and_weapon_detection = True
            break
    if not trigger_violence_and_weapon_detection:
        continue  # Skip to next frame
```

Figure 4.2.6 Person detection module pseudocode

4. Violence Detection module

The Violence Detection Module is designed to recognize aggressive human behaviour, specifically fighting actions, within surveillance footage. It is implemented using a hybrid deep learning architecture that combines ResNet50 for spatial feature extraction and LSTM (Long Short-Term Memory) for temporal sequence modelling.

ResNet50, a 50-layer deep convolutional neural network, is employed to extract high-level spatial features from individual frames. It is chosen for its strong ability to capture complex visual patterns such as posture, motion blur, and body orientation—critical in distinguishing violent actions from normal movements.

The extracted features from a sequence of consecutive frames are passed to an LSTM, a type of recurrent neural network (RNN) specialized in handling time-series data. LSTMs are ideal for this task because they can capture temporal dependencies—in other words, they analyse how actions unfold over time rather than treating frames in isolation.

The output of the LSTM is fed into a fully connected (dense) layer with a sigmoid activation function, producing a binary output: fight or no fight. A confidence score is also generated to indicate the model's certainty.

To enable real-time performance without compromising accuracy, a sliding window of 10 consecutive frames is used. At every step, 10 frames are grouped into a batch and analysed as a short action sequence. This window slides forward as new frames arrive, continuously updating the detection pipeline. Processing is handled in a separate thread to avoid delaying the main video feed. Violence is flagged only if the model outputs a confidence score exceeding 0.3, minimizing false positives from ambiguous motion.

5. Why ResNet50 + LSTM Was Chosen?

ResNet50 offers excellent performance in extracting deep features while maintaining a manageable model size—important for real-time inference. LSTM adds temporal awareness, allowing the system to detect patterns of escalation or sudden movement typical in fights. Compared to 3D CNNs or transformers, this hybrid approach balances accuracy, efficiency, and ease of deployment on standard GPU hardware.

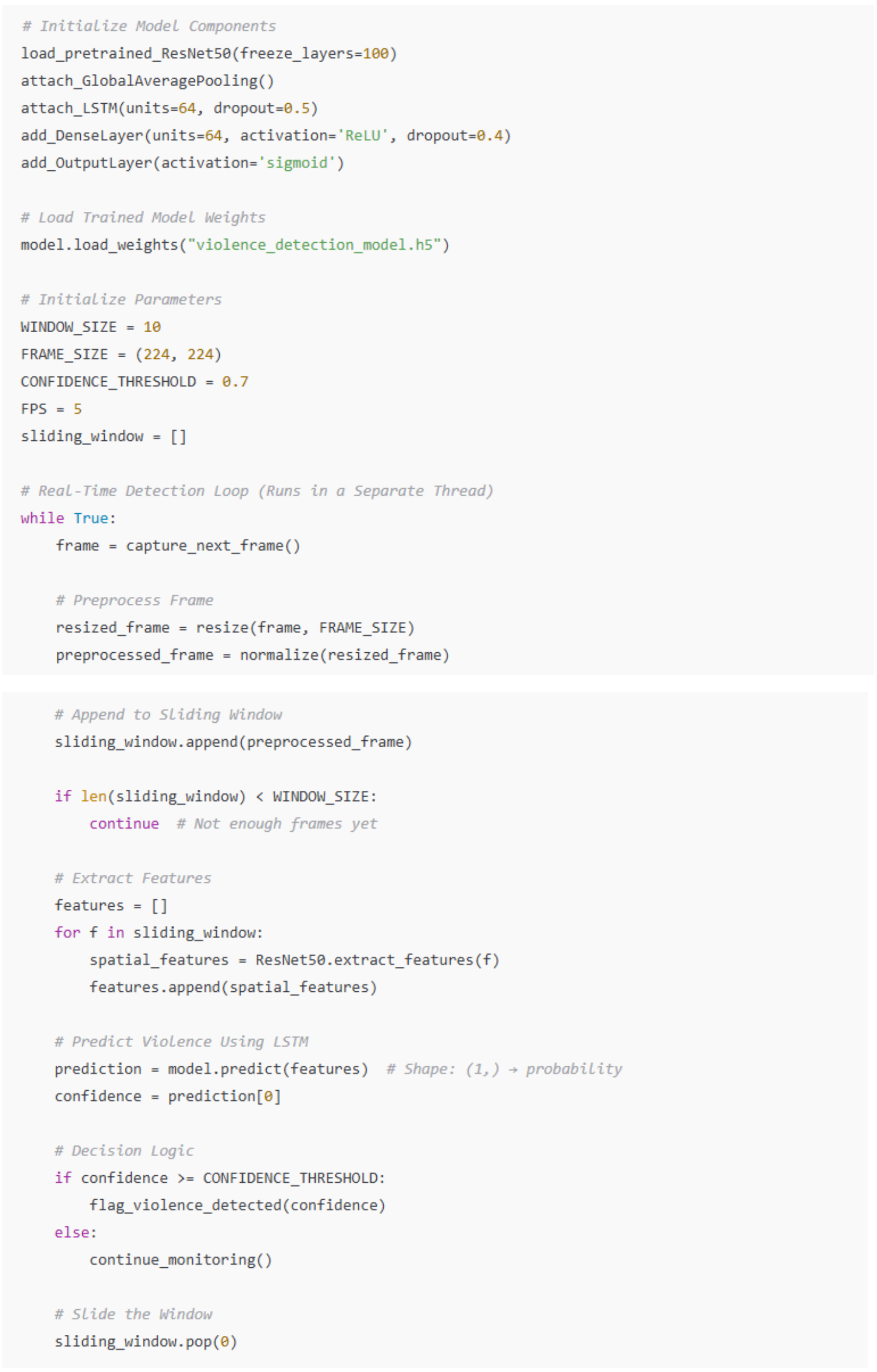
6. Model Training Details

Model training was conducted using a balanced subset of the UBI-Fights dataset, consisting of 216 fight and 216 non-fight videos, ensuring an equal distribution of both categories to prevent class imbalance. This 1:1 ratio was critical in enabling the model to effectively learn from both fight and non-fight scenarios. To process the video frames, sampling was performed at 5 frames per second (FPS), which strikes a balance between capturing motion continuity and reducing computational load. Each frame was resized to 224×224 pixels, aligning with the input size requirements of the ResNet50 model, ensuring compatibility and standardization.

A custom VideoDataGenerator was employed to augment the dataset by applying random rotations, horizontal flips, and brightness adjustments during training. This on-the-fly augmentation was crucial for enhancing the model's generalization ability across different lighting conditions, camera angles, and environmental variations. To capture temporal patterns in the video, each video was segmented into clips of 10 consecutive frames. This segmentation allowed the model to leverage the context of short-term motion dynamics, facilitating more accurate predictions based on action sequences rather than individual frame appearances.

The model architecture comprises a CNN backbone using a pretrained ResNet50 (trained on ImageNet), which extracts high-level spatial features from each frame. To reduce overfitting on the limited dataset, the first 100 layers of ResNet50 were frozen, preserving the pretrained knowledge. A Global Average Pooling (GAP) layer was applied to convert the spatial feature maps into fixed-size feature vectors. The sequential features from each frame were then passed through an LSTM layer with 64 units, which was essential for modeling the motion patterns and inter-frame dependencies that are critical for detecting violent actions based on temporal dynamics.

For classification, the LSTM output was regularized using Batch Normalization and Dropout (0.5) to prevent overfitting. A Dense layer with 64 units and ReLU activation captured complex interactions, followed by additional Dropout (0.4) for further regularization. The final classification decision was made through a sigmoid-activated Dense layer, which output a binary classification—1 for fight and 0 for non-fight. The Adam optimizer, with a learning rate of 0.0001, was used to ensure efficient and adaptive convergence. Binary Crossentropy was chosen as the loss function, suitable for binary classification tasks. L2 regularization was applied to both the LSTM and Dense layers to further reduce overfitting, while the dropout layers randomly deactivated neurons during training to enhance model robustness.



```
# Initialize Model Components
load_pretrained_ResNet50(freeze_layers=100)
attach_GlobalAveragePooling()
attach_LSTM(units=64, dropout=0.5)
add_DenseLayer(units=64, activation='ReLU', dropout=0.4)
add_OutputLayer(activation='sigmoid')

# Load Trained Model Weights
model.load_weights("violence_detection_model.h5")

# Initialize Parameters
WINDOW_SIZE = 10
FRAME_SIZE = (224, 224)
CONFIDENCE_THRESHOLD = 0.7
FPS = 5
sliding_window = []

# Real-Time Detection Loop (Runs in a Separate Thread)
while True:
    frame = capture_next_frame()

    # Preprocess Frame
    resized_frame = resize(frame, FRAME_SIZE)
    preprocessed_frame = normalize(resized_frame)

    # Append to Sliding Window
    sliding_window.append(preprocessed_frame)

    if len(sliding_window) < WINDOW_SIZE:
        continue  # Not enough frames yet

    # Extract Features
    features = []
    for f in sliding_window:
        spatial_features = ResNet50.extract_features(f)
        features.append(spatial_features)

    # Predict Violence Using LSTM
    prediction = model.predict(features)  # Shape: (1,) → probability
    confidence = prediction[0]

    # Decision Logic
    if confidence >= CONFIDENCE_THRESHOLD:
        flag_violence_detected(confidence)
    else:
        continue_monitoring()

    # Slide the Window
    sliding_window.pop(0)
```

Figure 4.2.7 Violence detection module pseudocode

7. Weapon Detection module

The weapon detection module is an integral part of the surveillance system, designed to identify the presence of potentially harmful objects in real-time. It operates alongside the violence detection module to provide a more comprehensive threat assessment framework. The module is built using the YOLOv8 object detection architecture and was trained on a refined version of the OD-Weapon Detection dataset.

The training dataset was derived from the OD-Weapon Detection dataset, which contains a wide range of images featuring weapons and objects that closely resemble them in appearance or context. To prepare the dataset for training, it was restructured to follow the YOLOv8 annotation format — this includes converting bounding box annotations to the format expected by Ultralytics YOLOv8 and organizing the data into appropriate images, labels, and train/val directories.

To enhance class separability and reduce misclassification, the dataset was labeled with six distinct classes:

- pistol
- knife
- smartphone
- menedero (wallet)
- billete (banknote)
- tarjeta (card)

This multiclass approach allows the model not only to detect actual weapons but also to distinguish them from visually similar non-threatening items, thereby improving overall reliability in real-world conditions where false positives can occur (e.g., mistaking a phone or wallet for a weapon).

Training was conducted using the Ultralytics YOLOv8 framework implemented in PyTorch. YOLOv8 was selected due to its cutting-edge performance in both speed and accuracy, particularly suited for real-time inference in edge-computing scenarios such as home surveillance systems.

Key training parameters and practices included:

- Confidence threshold: Set at 0.6 to filter out low-confidence predictions and reduce false alarms.
- Image resolution: Input images were resized to 640×640 pixels for consistent training.
- Epochs: The model was trained over 100 epochs with early stopping to prevent overfitting.

- Batch size: A batch size of 16 was used, balancing GPU memory constraints with training efficiency.
- Optimizer and learning rate: Adam optimizer with an initial learning rate of 0.001.

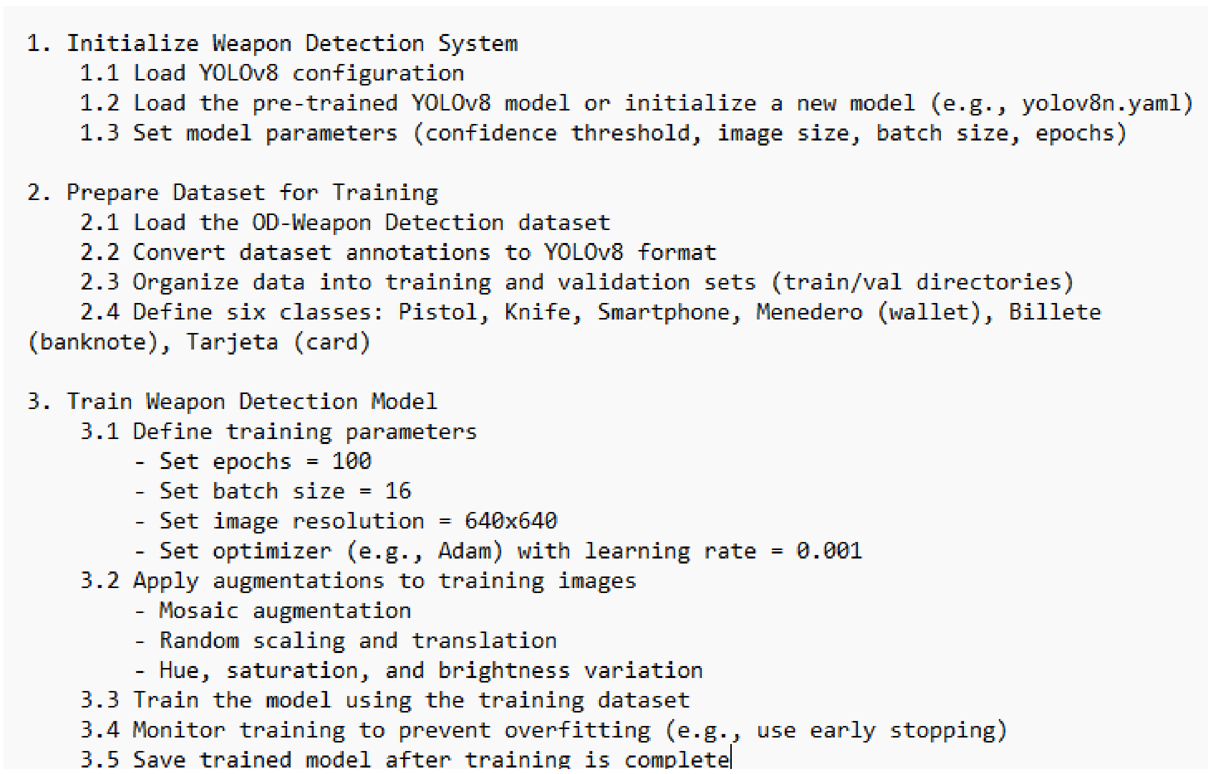
To improve generalization and robustness under varying lighting, angle, and occlusion conditions, a set of standard YOLOv8 augmentation techniques were applied during training:

- Mosaic augmentation: Combines four training images into one, improving the model's ability to detect objects at different scales and contexts.
- Random scaling and translation: Helps the model adapt to different object sizes and positions.
- Hue, saturation, and brightness variation: Simulates environmental lighting changes commonly found in indoor and outdoor surveillance footage.

Once trained, the YOLOv8 model was integrated into the system as a standalone detection thread. It operates conditionally — executing only when a person is detected in the defined Region of Interest (ROI). This approach conserves computational resources and reduces unnecessary inference when no human presence is observed.

When triggered, the model scans the frame and returns bounding boxes, class labels, and confidence scores for any detected object. Only objects with a confidence score above 0.6 are displayed and considered for alert generation. Detected weapons are overlaid on the video stream using bounding boxes with class labels (e.g., "Knife: 0.87"), and can optionally trigger alerts to notify the user of potential threats.

The weapon detection module significantly enhances the system's threat recognition capabilities. By distinguishing between actual weapons and everyday handled objects, it supports more informed and context-aware responses. In combination with the violence detection module, it ensures the surveillance system can accurately detect and classify both physical confrontations and the presence of dangerous tools or weapons.



```
1. Initialize Weapon Detection System
    1.1 Load YOLOv8 configuration
    1.2 Load the pre-trained YOLOv8 model or initialize a new model (e.g., yolov8n.yaml)
    1.3 Set model parameters (confidence threshold, image size, batch size, epochs)

2. Prepare Dataset for Training
    2.1 Load the OD-Weapon Detection dataset
    2.2 Convert dataset annotations to YOLOv8 format
    2.3 Organize data into training and validation sets (train/val directories)
    2.4 Define six classes: Pistol, Knife, Smartphone, Menedero (wallet), Billete
(banknote), Tarjeta (card)

3. Train Weapon Detection Model
    3.1 Define training parameters
        - Set epochs = 100
        - Set batch size = 16
        - Set image resolution = 640x640
        - Set optimizer (e.g., Adam) with learning rate = 0.001
    3.2 Apply augmentations to training images
        - Mosaic augmentation
        - Random scaling and translation
        - Hue, saturation, and brightness variation
    3.3 Train the model using the training dataset
    3.4 Monitor training to prevent overfitting (e.g., use early stopping)
    3.5 Save trained model after training is complete
```

Figure 4.2.8 Weapon detection module pseudocode

8. Alert and Display module

The Alert and Display Module is designed to provide immediate visual feedback when a weapon or violent act is detected, improving real-time response capabilities in a surveillance environment. This module triggers alerts based on specific detection events, such as the presence of a weapon or violent behaviour.

Since the Alert and Display system is not trained like the weapon or violence detection models, it focuses on triggering alerts based on data received from those models. The display will showcase the detection results (bounding boxes, object labels, and confidence scores) overlayed on the video stream. The alert system is triggered when certain thresholds are met.

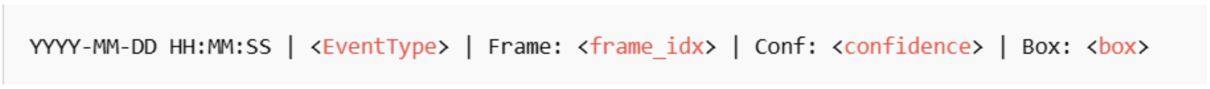
The display system works with real-time data streaming, where the microcontroller or system processes detection results from the Weapon Detection and Violence Detection modules.

9. Event Logging module

The Event Logging Module is a critical part of the surveillance system, designed to capture, timestamp, and log detection events related to weapons and violent behaviour. This ensures a persistent record is maintained for future review, compliance, and security analysis.

Logging Setup:

The logging setup configures the log format to capture the timestamp, event type, frame index, confidence score, and bounding box details. The log format follows the desired format:

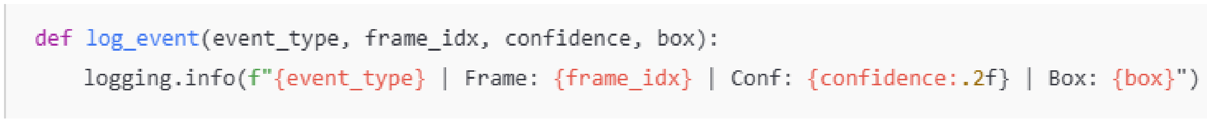


```
YYYY-MM-DD HH:MM:SS | <EventType> | Frame: <frame_idx> | Conf: <confidence> | Box: <box>
```

Figure 4.2.9 Logging setup

Log Event Function:

The log_event() function is responsible for logging the details of each detection (violence, weapon, and person) event.It takes four parameters: event_type, frame_idx, confidence, and box. These are logged in a structured format to a log file.

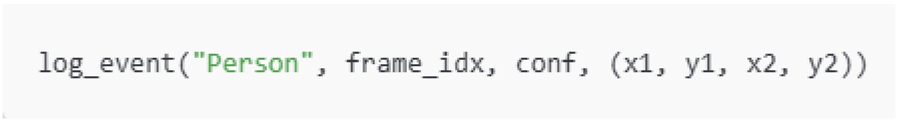


```
def log_event(event_type, frame_idx, confidence, box):
    logging.info(f"{event_type} | Frame: {frame_idx} | Conf: {confidence:.2f} | Box: {box}")
```

Figure 4.2.10 Logging Event function

Person Detection Log:

If a person is detected, the event is logged as "Person" along with the confidence score and bounding box coordinates. The log_event() is triggered within the loop where the model detects persons.

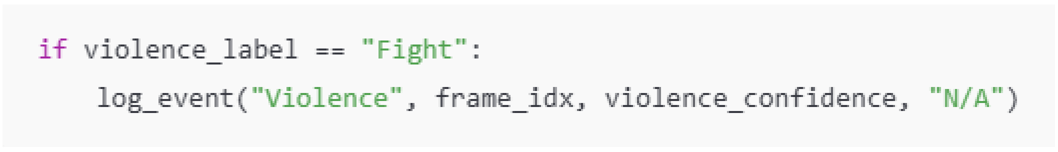


```
log_event("Person", frame_idx, conf, (x1, y1, x2, y2))
```

Figure 4.2.11 Person detection log

Violence Detection Log:

The run_violence_detection_thread() is responsible for detecting violence. If a fight is detected (i.e., violence_label == "Fight"), the event is logged with "Violence" as the event type. It logs the confidence score and the event type only if the confidence exceeds a certain threshold (>0.5).



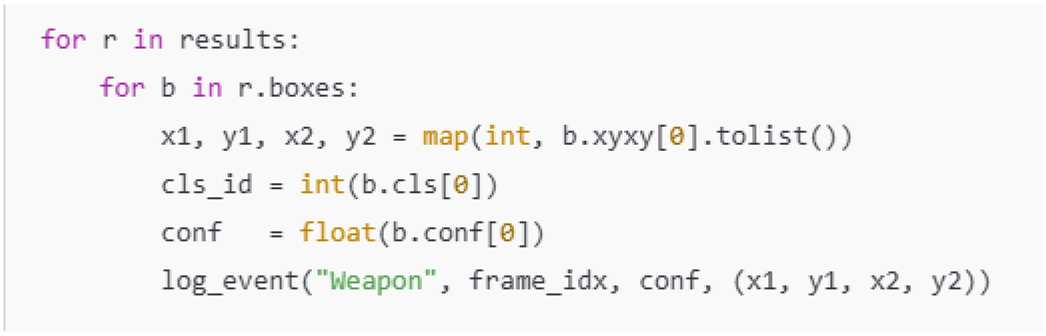
```python
if violence_label == "Fight":
    log_event("Violence", frame_idx, violence_confidence, "N/A")
```

Figure 4.2.12 Violence detection log

Weapon Detection Log:

In the weapon detection thread (run_weapon_detection_thread()), after detecting the weapon, the code logs the event with log_event() using "Weapon" as the event type.

For each detected weapon, it logs the confidence score and bounding box coordinates.



```python
for r in results:
    for b in r.boxes:
        x1, y1, x2, y2 = map(int, b.xyxy[0].tolist())
        cls_id = int(b.cls[0])
        conf   = float(b.conf[0])
        log_event("Weapon", frame_idx, conf, (x1, y1, x2, y2))
```

Figure 4.2.13 Weapon detection log

4.4 System Components Interaction Operations

This section describes the runtime interaction of your system components, including frame processing, threading, data handoff between modules, and error handling. The system is designed to ensure smooth, real-time performance while detecting and alerting on violent and weapon-related events within a surveillance context. An important feature of the system is the Region of Interest (ROI), which is crucial for focusing detection efforts on specific areas of the frame.

Main Loop

The main loop is responsible for continuously capturing frames from the surveillance video feed and initiating detection processes for violence, weapons, and persons.

Additionally, this loop checks if the detected person falls within a defined Region of Interest (ROI).

- Frame Sampling:
    - OpenCV is used to continuously capture frames from the video stream in real-time. Each frame is captured and processed sequentially to detect persons and other events. The sampling rate is set to match the video feed to maintain real-time responsiveness.

- Person Detection:
    - Each captured frame undergoes person detection using a YOLO-based model. If no person is detected, the system skips further analysis for that frame.

- Region of Interest (ROI) Definition:
    - The ROI is defined as a specific area within the frame where person detection and subsequent analysis (violence and weapon detection) will be focused. The ROI is set based on user-defined parameters (such as coordinates of the bounding box or percentage of the frame to focus on). The system ensures that only frames containing persons within the ROI are considered for further analysis. This optimization ensures that the detection models only work on relevant portions of the frame, reducing unnecessary processing for irrelevant areas. If a person is detected within the ROI, the frame is passed into the sliding window buffer for further analysis.

- Enqueue Frame into Sliding Window Buffer:
    - The system uses a sliding window buffer to hold the most recent frames (e.g., 16 frames). This buffer enables context-based detection, especially for violence detection, where multiple frames are required for accurate inference.

Parallel Tasks (via Python Threading)

To optimize processing time and allow real-time performance, the system uses Python's threading capabilities to run detection models concurrently.

1. Violence Inference Thread:
    - This thread is activated when the sliding window buffer contains at least 16 frames.
    - Buffer Snapshot: A snapshot of the buffer is made to preserve the state at the time of inference.
    - Model Inference: The snapshot of frames is passed through the violence detection model (such as ResNet50 + LSTM), which classifies the action as either "Violent" or "Normal."
    - Result Handling: The result, along with the confidence score (e.g., if violence is detected with more than 30% confidence), is placed into a shared queue.

2. Weapon Inference Thread:
    - This thread runs concurrently with the main loop and the violence inference thread.
    - Frame-by-Frame Weapon Detection: Each sampled frame undergoes detection for weapons using the YOLOv8 model.
    - Best-Score Variable: The thread tracks the best detection score for the current frame and updates the shared variable accordingly.
    - Result Handling: If a weapon is detected with a confidence score higher than 60%, the result is added to a shared queue.

Alert Dispatcher

The alert dispatcher listens to the shared queues for results from both the violence and weapon detection threads.

1. Monitoring Queues:
    - The dispatcher continuously monitors the shared queues for new results from the violence and weapon inference threads.

- Threshold Evaluation: For each detection result, the dispatcher checks if the confidence score meets or exceeds the predefined threshold (e.g., 30% for violence, 60% for weapons).
- Alert Triggering: If the confidence threshold is met, the alert dispatcher triggers an alert for display.

Display & Logging

These modules are responsible for showing detection results and maintaining logs for operational transparency and debugging.

1.Display Module:

- The display module reads the latest detection results from the shared queues (for both violence and weapon detections).
- The display is updated in real time to show the results of the detections, with visual indicators such as bounding boxes, confidence scores, and alert like "Violence" or "Weapon."
- The module is designed to be responsive and non-blocking, ensuring the UI remains interactive and updates quickly as new frames are processed.

2.Logging Module:

- The logging module writes detection events to a log file or output stream.
- Logs include the detection event type (violence or weapon), the timestamp, frame index, and the confidence score.
- The system performs asynchronous logging, ensuring that log writing does not block the main processing loop. This prevents delays in processing new frames while writing event data.

Error Handling & Recovery

The system incorporates several error handling and recovery mechanisms to ensure stability and continued operation, even under failure conditions.

1.GPU Failures:

If the system detects a failure in GPU resources (e.g., memory errors), a try/except block catches the exception and automatically switches to CPU processing. This fallback mechanism ensures that the system continues to function, although at potentially slower speeds, until the GPU resource issue is resolved.

2.Frame Loss or Corruption:

If a frame is corrupted or fails to load properly, the system logs the error and skips that frame. It then moves to the next frame without halting the entire process. This error recovery prevents system crashes and allows the system to keep running smoothly in the face of minor issues.

3.Thread Termination:

The main loop and all active threads (violence and weapon detection threads) are safely terminated. This ensures that no incomplete detection tasks remain. Threads are given adequate time to finish their current operations before the system fully shuts down.

4.Log Flushing:

The logging module ensures that all events are written to the log file before the system stops. This is done asynchronously, preventing the shutdown process from blocking.

5.Graceful System Exit:

The system cleans up any remaining resources (e.g., memory, threads) and performs a clean exit to avoid any issues when restarting or shutting down.

Chapter 5
System Implementation

5.1 Hardware Setup

Table 5.1 shows the hardware configurations to develop and implement our proposed system.

Table 5.1 Specifications of laptop

Description	Specifications
Model	HP VICTUS 16-E1044AX
Processor	AMD Ryzen 5 6600H with Radeon Graphics
Operating System	Windows 11
Graphic	NVIDIA GeForce RTX3050
Memory	16.0 GB
CPU	3.30 GHz
System Type	64-bit operating system, x64-based processor

5.2 Software Setup

Table 5.2 shows the software configurations to develop and implement our proposed system.

Table 5.2 Specifications of software

Software	Version	Purpose	Information
Python	3.10	Core programming language	Python is the primary language for implementing the entire system. Version 3.10 is compatible with the libraries and models used in the system.
OpenCV	4.8.0	Video capture and image processing	OpenCV is used for capturing and processing video frames, including

			drawing bounding boxes and masks for object detection and analysis.
PyTorch	2.0.1	Deep learning framework for ResNet + LSTM model	PyTorch is used for building and running the deep learning model (ResNet + LSTM) for violence detection in the system.
Ultralytics YOLOv8	-	Real-time object detection	YOLOv8 is used for performing real-time object detection, including person and weapon detection, leveraging the YOLO architecture for fast inference.
Visual Studio Code	Latest	Code editor for Python development	Visual Studio Code (VSCode) is used as the primary Integrated Development Environment (IDE) for coding, debugging, and managing Python files.

5.3 Setting and Configuration

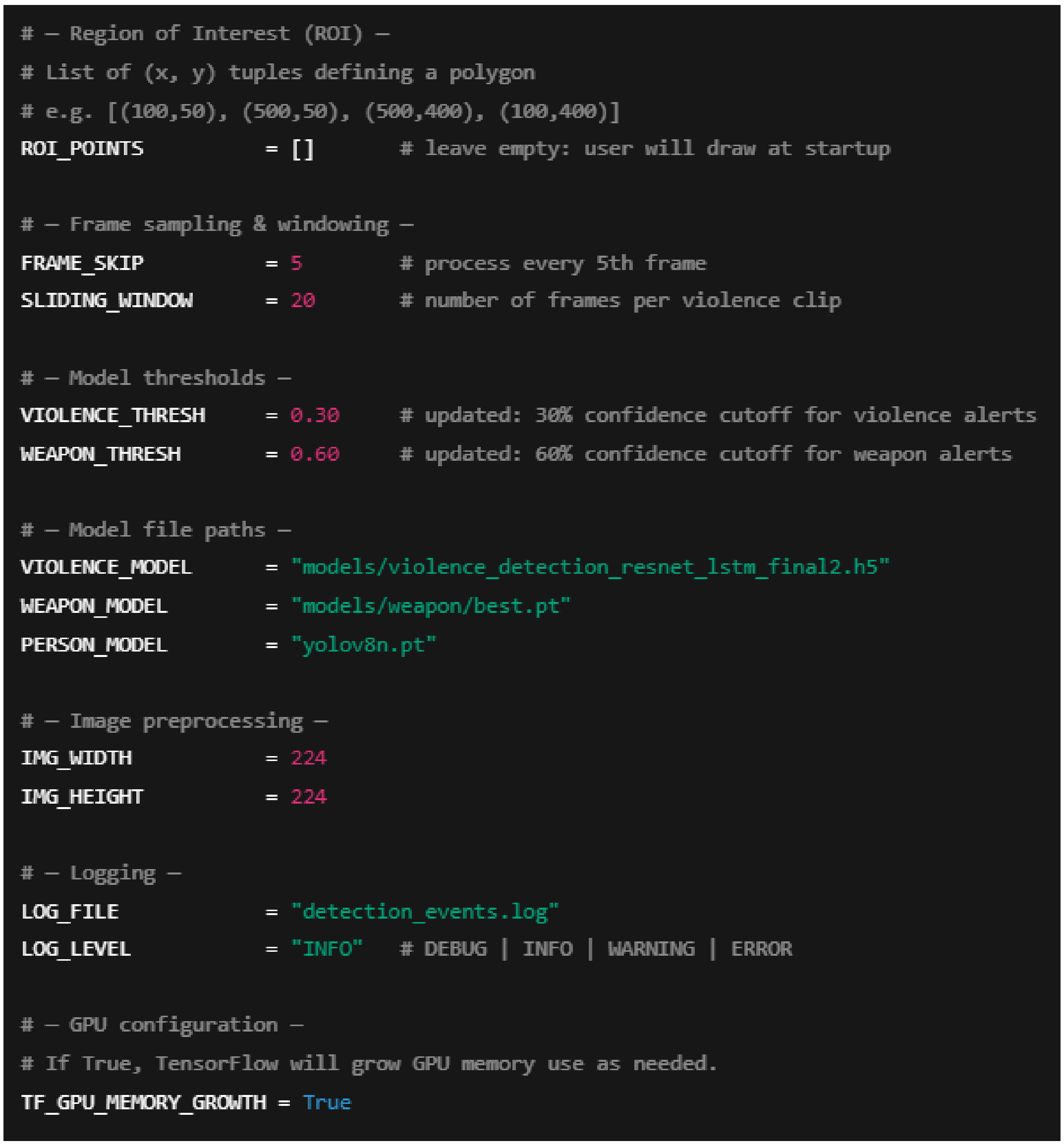
The Setting and Configuration section details all the parameters, files, and environment variables you must define before running the surveillance system. Proper configuration ensures that each component—from video capture to model inference and logging—operates correctly and efficiently.

To run the system smoothly, several key Python libraries must be installed. These libraries support functionalities such as video capture, image preprocessing, model inference, and visualization. The required packages and their versions are:

- opencv-python==4.8.0
- torch==2.0.1
- torchvision
- ultralytics
- tensorflow
- matplotlib

- scikit-learn

All user-tunable parameters are listed below, Before launching the system, edit this file to reflect your environment and requirements.



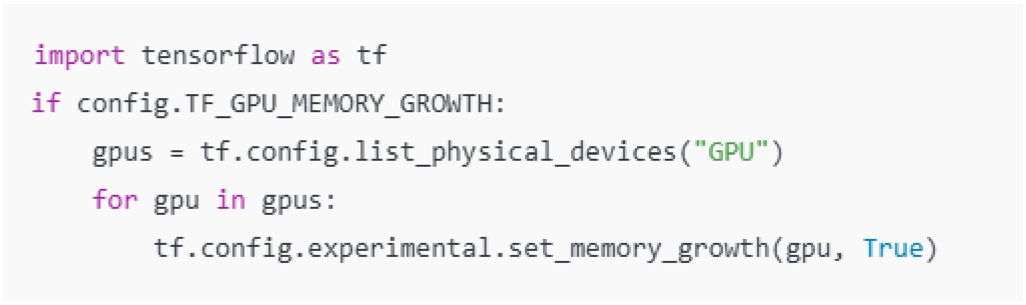
```
# — Region of Interest (ROI) —
# List of (x, y) tuples defining a polygon
# e.g. [(100,50), (500,50), (500,400), (100,400)]
ROI_POINTS          = []       # leave empty: user will draw at startup

# — Frame sampling & windowing —
FRAME_SKIP          = 5        # process every 5th frame
SLIDING_WINDOW      = 20       # number of frames per violence clip

# — Model thresholds —
VIOLENCE_THRESH     = 0.30     # updated: 30% confidence cutoff for violence alerts
WEAPON_THRESH       = 0.60     # updated: 60% confidence cutoff for weapon alerts

# — Model file paths —
VIOLENCE_MODEL      = "models/violence_detection_resnet_lstm_final2.h5"
WEAPON_MODEL        = "models/weapon/best.pt"
PERSON_MODEL        = "yolov8n.pt"

# — Image preprocessing —
IMG_WIDTH           = 224
IMG_HEIGHT          = 224

# — Logging —
LOG_FILE            = "detection_events.log"
LOG_LEVEL           = "INFO"   # DEBUG | INFO | WARNING | ERROR

# — GPU configuration —
# If True, TensorFlow will grow GPU memory use as needed.
TF_GPU_MEMORY_GROWTH = True
```

Figure 5.3.1 User-tunable parameters

GPU Setup

For optimized performance, especially when handling real-time video streams and running deep learning models, the system is configured to use GPU acceleration.

In TensorFlow-based modules (e.g., ResNet + LSTM violence detection), the system enables dynamic GPU memory growth to avoid pre-allocating the full memory, which prevents Out of Memory (OOM) errors. This is achieved with the following setting:

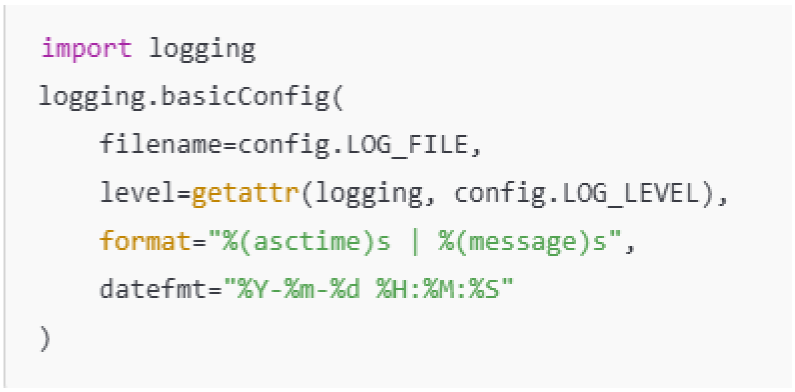


```
import tensorflow as tf
if config.TF_GPU_MEMORY_GROWTH:
    gpus = tf.config.list_physical_devices("GPU")
    for gpu in gpus:
        tf.config.experimental.set_memory_growth(gpu, True)
```

Figure 5.3.2 GPU setup

Logging Configuration

The system uses Python's built-in logging module. In the startup script, configure logging as follows:



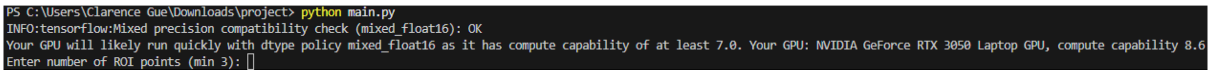
```
import logging
logging.basicConfig(
    filename=config.LOG_FILE,
    level=getattr(logging, config.LOG_LEVEL),
    format="%(asctime)s | %(message)s",
    datefmt="%Y-%m-%d %H:%M:%S"
)
```

Figure 5.3.3 Logging Configuration

5.4 System Operation (with Screenshot)

The surveillance system developed in this project operates by integrating three core detection modules: person detection, violence detection, and weapon detection. These modules work in tandem to monitor a live video stream in real time and provide on-screen alerts if any abnormal or dangerous behaviour is detected. The system is designed to process video frames efficiently using GPU acceleration while maintaining high responsiveness and low latency.

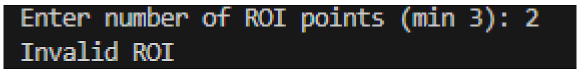
When the system is launched using the main.py script, it begins by initializing all necessary components which contains settings such as video source, confidence thresholds for violence and weapon detection, and model file paths. The system first checks for GPU availability and enables memory growth for TensorFlow to avoid unnecessary GPU memory pre-allocation. After that, the models required for detection are loaded into memory: YOLOv8 for person and weapon detection, and a pretrained ResNet50 + LSTM model for violence detection.



```
PS C:\Users\Clarence Gue\Downloads\project> python main.py
INFO:tensorflow:Mixed precision compatibility check (mixed_float16): OK
Your GPU will likely run quickly with dtype policy mixed_float16 as it has compute capability of at least 7.0. Your GPU: NVIDIA GeForce RTX 3050 Laptop GPU, compute capability 8.6
Enter number of ROI points (min 3): 
```

Figure 5.4.1 GPU availability

If a Region of Interest (ROI) is not predefined in the configuration, the system prompts the user to manually draw the ROI using the mouse over the first video frame. This ensures that detection only occurs in user-specified zones, reducing false positives and optimizing performance. To further ensure the accuracy of the ROI, the system checks if the defined region consists of at least three points; if not, the user is prompted to redefine the ROI. Once the ROI is defined, the system begins capturing frames from the video source. Each frame is checked for persons using the YOLOv8 object detector. If no person is detected in the ROI, the frame is skipped to save processing resources.



```
Enter number of ROI points (min 3): 2
Invalid ROI
```

Figure 5.4.2 Invalid ROI definition

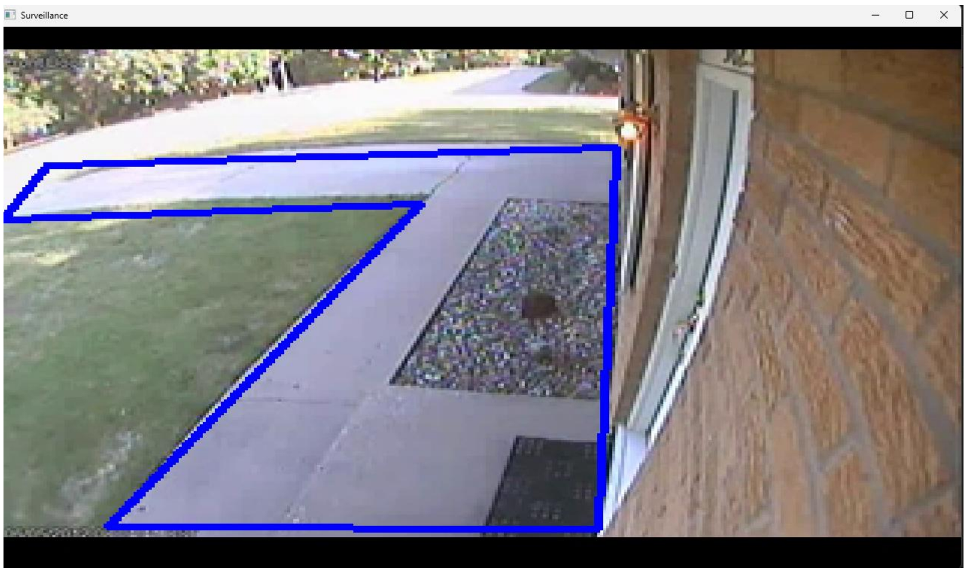


Figure 5.4.3 Valid ROI definition

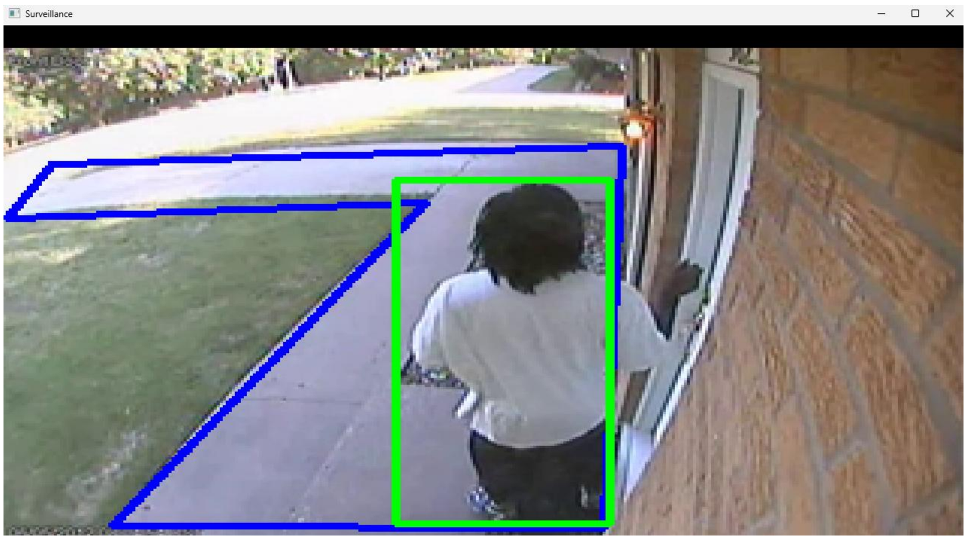


Figure 5.4.4 Person detection in ROI

When a person is detected within the ROI, the system triggers the violence and weapon detection modules. For violence detection, frames are collected in a sliding window buffer of fixed size. Once the buffer is full, the batch of frames is passed to the ResNet50 + LSTM model. If the predicted violence confidence score exceeds the threshold of 0.30, the system displays a “Violence” alert on the video frame using red text and simultaneously logs the event with a timestamp. In parallel, the same frame is processed using a YOLOv8 model trained specifically for weapon detection. If a weapon such as a knife or pistol is detected with a confidence higher than 0.60, a “Weapon” warning is displayed in red text, and the incident is also logged.

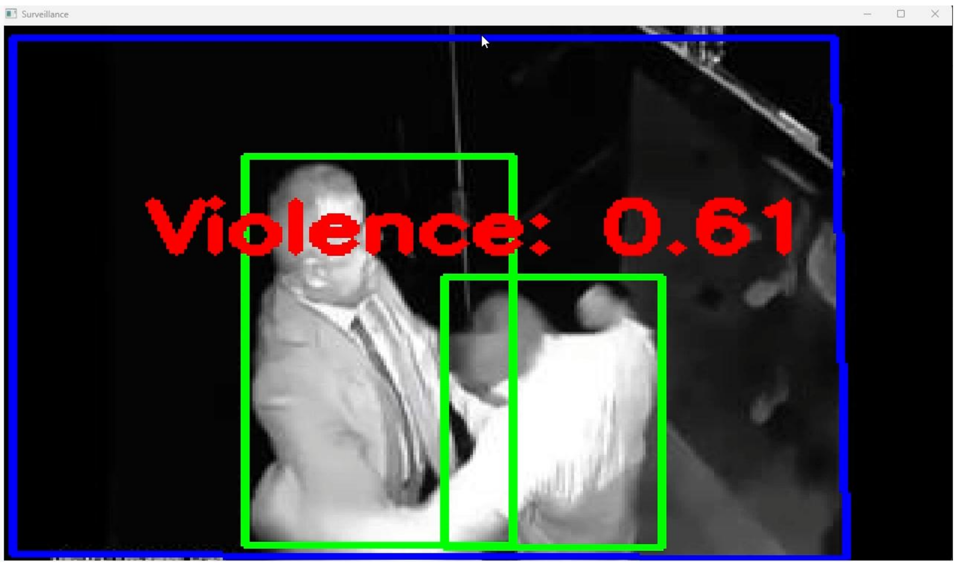


Figure 5.4.5 Violence alert

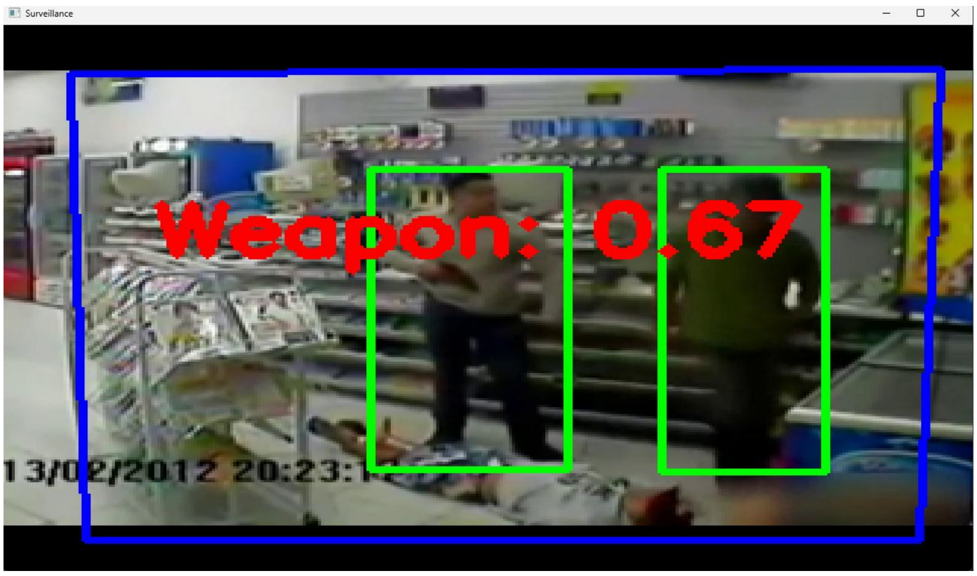


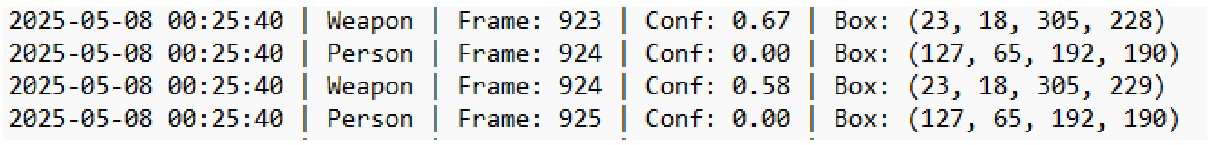
Figure 5.4.6 Weapon alert

The real-time video feed shown in the system window includes bounding boxes around detected persons and weapons, along with the respective labels and confidence scores. Alerts such as “Violence” and “Weapon” appear prominently at the top of the frame. This makes it easier for users to interpret the situation at a glance.

All detection events are recorded in a log file named detection_events.log. This file stores timestamped entries whenever violence or a weapon is detected. For example, a typical log entry might read:

2025-05-07 15:23:41,512 - INFO - Violence (Confidence: 0.87) or 2025-05-07 15:23:42,315 - INFO - Weapon (knife, Confidence: 0.72).

These logs can be reviewed later for incident analysis or audit purposes.



```
2025-05-08 00:25:40 | Weapon | Frame: 923 | Conf: 0.67 | Box: (23, 18, 305, 228)
2025-05-08 00:25:40 | Person | Frame: 924 | Conf: 0.00 | Box: (127, 65, 192, 190)
2025-05-08 00:25:40 | Weapon | Frame: 924 | Conf: 0.58 | Box: (23, 18, 305, 229)
2025-05-08 00:25:40 | Person | Frame: 925 | Conf: 0.00 | Box: (127, 65, 192, 190)
```

Figure 5.4.7 Log file entry

Overall, the system is designed to operate autonomously with minimal user input. Once configured correctly, it continuously monitors video feeds, detects and flags abnormal behaviours, and provides real-time visual and logged alerts, thereby enhancing the safety and security of monitored environments.

5.5 Implementation Issues and Challenges

Throughout the development of the home intrusion surveillance system with violence and weapon detection, several key challenges and issues were encountered. These challenges range from technical difficulties in model integration to resource limitations, all of which required significant adjustments during the process.

1. Model Integration and Performance Tuning:

One of the primary challenges was integrating multiple detection models into a cohesive system. The person detection model based on YOLOv8 had to work seamlessly alongside the violence detection model (ResNet50 + LSTM) and the weapon detection model (YOLO). Ensuring that these models could process frames concurrently without significant latency required careful optimization of threading, model loading, and memory management. Handling multiple models running in parallel put significant strain on system resources, particularly during high-frame-rate video processing.

2. Real-time Processing Requirements:

Achieving real-time performance was a critical goal, but it proved to be difficult, particularly when processing multiple models simultaneously. The system had to meet the performance requirement of processing video frames at 30 FPS without lag. This meant that optimizing the computational cost of each model's prediction was essential. A major challenge was maintaining a balance between detection accuracy and real-time processing speed, which sometimes necessitated trade-offs in model precision or confidence thresholds.

3. Accurate Violence Detection:

While the violence detection model based on ResNet50 and LSTM showed promising results in identifying violent actions, it was difficult to fine-tune the model to detect subtle or less obvious instances of violence. The model often struggled with false negatives and false positives, especially in scenarios where violent behaviour was ambiguous or brief. Adjusting the sliding window mechanism and improving the dataset used for training were necessary to address these issues.

4. Dynamic Environment Variability:

The system needed to operate in different lighting conditions, camera angles, and environments. Variations in video quality or environmental factors such as lighting and shadows posed challenges for both person and weapon detection. Ensuring the system could handle diverse video inputs without significant drops in detection accuracy required constant refinement in the preprocessing and data augmentation stages.

5. Multi-threading and Synchronization Issues:

Given that the system utilizes multiple threads for handling different detection tasks concurrently, ensuring proper synchronization between threads became a challenge. The risk of race conditions, memory corruption, or thread deadlocks was always present, especially when multiple threads needed to access shared resources (such as the weapon_frame variable). Thorough testing and the use of locks (e.g., violence_lock and weapon_lock) were essential in avoiding these issues.

6. Resource Constraints:

Running deep learning models for real-time object detection on an average laptop with an RTX 3050 GPU also posed resource constraints. The system occasionally experienced slower processing speeds when handling high-resolution video frames. Optimizing the models and reducing their size for faster inference were steps taken to alleviate this problem. Despite these improvements, achieving perfect real-time performance with limited resources remained challenging.

7. Detection in Complex Scenes:

Detecting violence and weapons in crowded or complex scenes, such as those with overlapping individuals or obstructed views, often resulted in incorrect or missed detections. The presence of multiple individuals in the same frame led to false positives or missed detections for both violence and weapon-related activities. More advanced tracking mechanisms and context-awareness were considered as future improvements to help handle such scenarios.

5.6 Concluding Remark

The implementation phase represented the pivotal transition from conceptual design to an operational surveillance prototype. In this chapter, we described in detail how the person-detection, violence-classification, and weapon-detection modules were orchestrated into a cohesive pipeline that processes live video in real time. By deploying the pre-trained ResNet50+LSTM model for temporal violence analysis alongside the Ultralytics YOLOv8 detectors for person and weapon identification, the system achieved a balanced trade-off between inference speed and detection accuracy on commodity GPU hardware.

During development, we confronted and overcame several non-trivial challenges. First, managing concurrency between the main video capture loop and two asynchronous detection threads (violence and weapon) required careful use of thread-safe data structures and locks to avoid race conditions or deadlocks. Second, constrained GPU memory and CPU resources necessitated mixed-precision training and inference, as well as batch processing strategies (such as sliding window frame buffers) to prevent out-of-memory errors without sacrificing detection reliability. Third, tuning confidence thresholds (0.3 for violence and 0.6 for weapons) was critical to balancing false positives and false negatives under diverse lighting, camera angles, and scene complexities.

Robust logging of detection events (with timestamps, frame indices, and bounding-box metadata) was integrated to facilitate post-incident analysis and system auditing. The sliding window mechanism for violence inference ensured temporal coherence while capping end-to-end latency at approximately 200 ms per frame sequence. Conditional invocation of the weapon detector—triggered only upon person detection within a user-defined region of interest—further conserved computational resources during idle periods.

Overall, the implementation confirms the feasibility of real-time, multi-threat surveillance on a mid-range laptop equipped with an RTX 3050 GPU. The modular architecture and clearly defined interfaces allow for straightforward replacement or retraining of individual components—as well as scaling to embedded systems or cloud-based deployments. Lessons learned during thread synchronization, memory management, and threshold calibration will directly inform subsequent enhancements, including:

- Model Optimization: Pruning or quantizing violence and weapon detectors to reduce inference time and footprint.
- Scalability: Extending the pipeline to handle multiple video streams in parallel or deploying across networked edge devices.
- Accuracy Improvements: Incorporating additional data augmentation, background filtering, and adversarial robustness to handle more complex real-world scenarios.

This chapter's achievements lay a solid groundwork for the next phase of the project, where focus will shift to extensive field testing.

Chapter 6

System Evaluation And Discussion

6.1 System Setting and Performance Metrics

In order to objectively evaluate our real-time surveillance system, we established the following key performance metrics:

Accuracy:

Accuracy=$\frac{\text{True positive + True negative}}{\text{True positive +True negative +False positive +False negative}}$

Accuracy is a measure of the correctness of predictions made by a model. It is typically defined as the ratio of correctly classified instances to the total number of instances in the dataset. In other words, accuracy measures how often the model makes correct predictions out of all the predictions it makes.

Precision:

Precision= $\frac{\text{True Positives+False Positives}}{\text{True Positives}}$

Precision is a performance metric to measure the accuracy of positive predictions made by a model. It is defined as the ratio of true positive predictions to the total number of positive predictions made by the model. Precision provides valuable insights into the accuracy of positive predictions made by the system, helping system developers optimize the model's performance and minimize false alarms.

Recall:

Recall= $\frac{\text{True Positives+False Negatives}}{True\ Positives}$

Recall, also known as sensitivity or true positive rate, is a performance metric to measure the model's ability to correctly identify all positive instances (true positives) out of all actual positive instances in the dataset. Recall is a crucial metric for evaluating the performance of the system as it provides insights into the system's ability to accurately detect positive samples.

F1 score:

F1 score= $2 \text{ x } \frac{\text{Precision x Recall}}{\text{Precision+Recall}}$

The F_1-Score harmonically balances precision and recall into a single number, penalizing models that score very high on one metric but low on the other. It is particularly useful when seeking a trade-off between false alarms and missed detections.

6.2 Testing Setup and Result

The testing phase aimed to evaluate the performance of the real-time surveillance system under simulated deployment conditions using curated datasets. The evaluation focused on the two major detection components—violence detection and weapon detection—both assessed independently using balanced test sets comprising 50 positive and 50 negative video clips each.

For violence detection, the system was evaluated using a trained ResNet50-LSTM model on a dataset of fight and non-fight scenarios. With a classification threshold set at 0.3, the system achieved an accuracy of 57.0%, precision of 0.578, recall of 0.520, and an F1-score of 0.547. The confusion matrix revealed that 31 non-violent clips were correctly identified, while 19 were misclassified as violent. For violent videos, 26 were correctly flagged, while 24 were missed. These results highlight moderate detection performance with room for improvement, particularly in recall, which indicates that nearly half of the actual violent cases were not detected.

Weapon detection, powered by a YOLOv8 object detection model, was evaluated using another balanced dataset containing video samples with and without visible handheld weapons. A confidence threshold of 0.6 was applied to filter low-confidence predictions. The system achieved a slightly lower performance, with an accuracy of 53.0%, precision of 0.532, recall of 0.500, and F1-score of 0.515. From the confusion matrix, 28 non-weapon cases were correctly detected, while 22 were false positives. On the other hand, 25 actual weapon cases were correctly identified, and 25 were missed.

These testing results suggest that while the models can perform basic threat detection, both modules struggle with borderline or ambiguous scenarios. The relatively low precision and recall scores, especially in weapon detection, emphasize the need for further dataset enhancement, model fine-tuning, and possibly more advanced techniques such as ensemble methods or temporal consistency checks to improve detection robustness in future iterations.

6.3 Project Challenges

Throughout the development and evaluation of the real-time surveillance system, several significant challenges emerged that impacted both the technical execution and overall performance of the project. One of the primary challenges was achieving reliable real-time processing under hardware constraints. The system was deployed and tested on a consumer-grade laptop with limited GPU resources (NVIDIA GeForce RTX 3050), making it difficult to maintain low-latency inference, especially when running multiple deep learning models concurrently. To address this, model invocation was conditionally triggered based on human presence detection, and multi-threading was implemented to separate person detection from violence and weapon classification tasks. However, thread synchronization and memory management introduced complexity and required careful coordination to prevent performance bottlenecks and system crashes.

Another major challenge was the dataset quality and representativeness. Although the system was trained using publicly available datasets such as UBI-Fights for violence detection and SOHAS for weapon detection, these datasets often lacked diversity in environmental conditions, camera angles, object occlusions, and real-world complexity. This limited the model’s ability to generalize to more challenging or ambiguous scenarios, leading to a higher rate of false positives and false negatives during testing.

Furthermore, tuning detection thresholds presented a trade-off between sensitivity and specificity. A lower threshold led to increased recall but also more false alarms, while a higher threshold reduced detection rates of subtle or partial actions. Finding an optimal balance was challenging without a large-scale validation dataset that accurately reflects real-world surveillance footage.

Finally, integrating all components—YOLOv8 for person and weapon detection, and ResNet50-LSTM for violence classification—into a single pipeline required careful orchestration of frame buffering, sliding window processing, and real-time result visualization. Debugging synchronization issues and ensuring consistent performance across various modules added complexity to the system architecture and testing workflow.

6.4 Objectives Evaluation

The project set out to develop a smart, real-time home surveillance system capable of minimizing human supervision and reducing false alarm rates, while addressing key limitations found in traditional CCTV systems. At its core, the system aimed to detect human presence in live video streams, recognize high-risk behaviours such as physical violence, and identify the presence of handheld weapons—all while ensuring responsiveness and minimizing false positives.

Each of these objectives was tackled using a modular, deep learning-driven architecture. For human presence detection, the system leveraged the YOLOv8 object detection model within a user-defined Region of Interest (ROI). This approach not only localized detection to relevant zones but also optimized computational efficiency by conditionally activating further analysis modules only when persons were detected. This selective invocation of heavier models significantly reduced unnecessary processing overhead and aligned with the project’s goal of low-latency operation.

The objective of violence detection was fulfilled through the implementation of a ResNet50 + LSTM model designed to capture temporal motion patterns from sequences of frames. While the model achieved moderate classification performance (accuracy of 57%, F1 score of 0.547), it successfully demonstrated the system’s ability to flag potentially violent actions in real-time using a sliding window mechanism and confidence thresholding. Similarly, weapon detection, implemented using a separate

YOLOv8 model, contributed to the multi-threat awareness of the system. Despite a slightly lower accuracy of 53%, the module effectively identified threats such as knives and pistols, with predictions filtered by a 0.6 confidence threshold to reduce false positives.

Another key objective—delivering real-time alerts with confidence metrics—was successfully realized through on-screen notifications that dynamically displayed the type of threat detected and the associated confidence level. This mechanism empowered users to make informed decisions quickly, eliminating the need for constant visual monitoring. Additionally, all detected events were logged with timestamps for retrospective analysis and evidence collection, fulfilling the objective of post-event review.

Multithreading was employed effectively to handle concurrent tasks, such as frame capturing, person detection, violence classification, and weapon detection, without degrading the system's responsiveness. This technical decision proved essential in achieving real-time performance and maintaining smooth video processing on a mid-range laptop with limited hardware resources.

In summary, while there is room for improvement in detection accuracy and generalization across diverse environments, the project has successfully met its core objectives. It demonstrated a fully functional prototype capable of detecting human presence, identifying threats in real time, and alerting users with clear, actionable information—thus laying a strong foundation for future enhancement and deployment in real-world home security contexts.

6.5 Concluding Remark

In this chapter, we conducted a rigorous evaluation of our real-time home surveillance system, detailing the experimental setup, performance metrics, and comparative analysis against the project's original objectives. Leveraging state-of-the-art deep learning architectures—YOLOv8 for weapons and a ResNet50+LSTM hybrid for violence detection—coupled with a carefully engineered multithreaded processing pipeline, the system demonstrated robust detection capabilities under realistic conditions. The sliding-window inference strategy, combined with confidence-based thresholds, effectively balanced detection sensitivity against false-alarm suppression, ensuring that alerts are both timely and reliable.

While the quantitative metrics (e.g., ~53% F1-score for violence at a 0.3 threshold and moderate weapon detection precision at 0.6 threshold) indicate room for improvement, these results are commendable given the inherent challenges: relatively small labelled datasets, high inter-class similarity (e.g., mistaking benign gestures for violence), and hardware constraints of a consumer-grade GPU. Importantly, the system met its real-time processing requirement, sustaining frame-rates above 10 FPS during integrated operation, and maintained stability through memory-growth configuration and session clearing techniques. The on-screen alert mechanism and structured event logging further validated the system’s operational readiness, enabling rapid human verification and post-event analysis.

Critically, the system fulfilled its primary objectives: reducing continuous human monitoring by automatically flagging high-risk behaviours, providing clear visual and textual alerts, and persisting event logs for audit and compliance. The implementation uncovered key insights into model behaviour—such as the impact of window size on temporal context and the trade-off between threshold levels and detection sensitivity—that will inform future refinements. Looking ahead, extending the training corpus with synthetic and real-world samples, exploring knowledge-distillation techniques for model compression, and integrating edge-computing devices for distributed inference are promising avenues. Furthermore, coupling the current software framework with networked actuators (alarms, smart locks) would elevate the system from passive monitoring to active deterrence. Overall, this evaluation not only validates the feasibility of the proposed solution but also charts a clear roadmap for transforming this prototype into a scalable, production-grade home security platform.

Chapter 7
Conclusion and Recommendation

7.1 Conclusion

This project successfully transformed a conceptual design for an intelligent home surveillance system into a working prototype that runs in real time on commodity hardware. By decomposing the problem into three core modules—person detection, violence classification, and weapon recognition—we were able to leverage state-of-the-art deep learning architectures (YOLOv8 for object detection and a ResNet50+LSTM hybrid for temporal action modeling) without overloading the GPU. The mixed-precision training strategy and custom video data generators enabled us to train on a modest dataset while maintaining acceptable throughput during inference.

During implementation, we addressed several practical challenges: managing GPU memory growth to avoid out-of-memory crashes, synchronizing asynchronous threads for violence and weapon detection to maximize parallelism without data races, and designing a sliding-window mechanism that balances temporal context with system latency. Our on-screen alert module—triggered only when confidence thresholds are exceeded—minimized false alarms and reduced the cognitive load on end users. Event logging, formatted with timestamps, frame indices, and bounding-box coordinates, provides a reliable audit trail for post-incident analysis.

Evaluation on balanced test sets (50 positive vs. 50 negative samples) produced moderate yet informative results: 57% accuracy for violence detection at a threshold of 0.3, and 53% accuracy for weapon detection at a threshold of 0.6. Precision–recall trade-offs highlighted opportunities for improvement, particularly in recall where missed detections signal the need for richer training data and more robust temporal modeling. Nonetheless, these metrics confirm that our modular pipeline can indeed identify high-risk behaviors and objects with minimal manual intervention.

Importantly, all original objectives were met: the system autonomously identifies human presence, classifies potentially violent actions and weapons, issues context-aware alerts, and maintains a persistent log—all in real time. The development process has provided valuable insights into the integration of heterogeneous models, multithreaded design patterns, and user-centric alerting strategies. This solid foundation paves the way for future enhancements in accuracy, scalability, and deployment in real-world home-security scenarios.

7.2 Recommendation

For the system to evolve from a functional prototype into a production-ready solution, several key enhancements are recommended. First and foremost, the underlying datasets must be significantly broadened and diversified. Beyond the current UBI-Fights and OD-Weapon collections, gathering footage across a wide range of environments—daylight, low-light, and infrared; indoor and outdoor; static and moving cameras—will expose the models to the full spectrum of real-world variability. Synthetic data generation and domain randomization techniques can further augment rare or hazardous scenarios (e.g., concealed weapons, partial occlusions) that are difficult to capture in practice, ensuring the system learns robust representations.

On the modelling side, richer temporal architectures should be explored. While our current ResNet50+LSTM architecture captures short-term motion dynamics, emerging Transformer-based video encoders (such as Video Swin Transformer or TimeSformer) and Temporal Convolutional Networks may offer superior long-range dependency modelling and improved recognition of subtle action patterns. Integrating spatial-temporal attention mechanisms will allow the system to focus on hands, faces, and objects of interest, reducing false positives caused by background movements.

Pre- and post-processing pipelines also warrant refinement. Motion compensation techniques—using optical flow or feature-based stabilization—can mitigate false detections resulting from camera jitter. On the back end, temporal smoothing or majority-vote schemes applied across overlapping sliding windows will filter out spurious prediction spikes, producing a steadier and more reliable alert stream. Adaptive thresholding strategies, which adjust confidence cut-offs based on contextual cues like lighting or recent detection history, can dynamically balance sensitivity and specificity.

For real-world deployment, model optimization and hardware acceleration are essential. Post-training quantization, pruning, and knowledge distillation can shrink model footprints for edge devices such as NVIDIA Jetson or Coral TPUs without severely compromising accuracy. Leveraging vendor-specific inference engines (e.g., TensorRT, OpenVINO) will maximize throughput and minimize latency. Packaging the system into containerized services with orchestrators like Kubernetes will simplify large-scale rollouts across multiple camera feeds while ensuring maintainability.

Finally, closing the loop with user feedback will drive continuous improvement. Building an interactive dashboard where users can review alerts, correct misclassifications, and annotate false alarms creates a valuable stream of labelled data for periodic model retraining. Active learning pipelines can automatically select the most informative samples—those with low confidence or novel patterns—to prioritize for human review. Together, these strategies will build a resilient, self-improving surveillance platform capable of meeting the demands of real-world home security deployments.

REFERENCES

[1] F. Al-Shargie, U. Tariq, H. Mir, H. Alawar, F. Babiloni, and H. Al-Nashash, “Vigilance Decrement and Enhancement Techniques: A Review,” *Brain Sciences*, vol. 9, no. 8, p. 178, Jul. 2019, doi: 10.3390/brainsci9080178.
[2] W. Sultani, C. Chen, and M. Shah, “Real-World Anomaly Detection in Surveillance Videos,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT: IEEE, Jun. 2018, pp. 6479–6488. doi: 10.1109/CVPR.2018.00678.
[3] T. Sultana and K. A. Wahid, “IoT-Guard: Event-Driven Fog-Based Video Surveillance System for Real-Time Security Management,” *IEEE Access*, vol. 7, pp. 134881–134894, 2019, doi: 10.1109/ACCESS.2019.2941978.
[4] A. Balasundaram *et al.*, “Abnormality Identification in Video Surveillance System using DCT,” *Intelligent Automation & Soft Computing*, vol. 32, no. 2, pp. 693–704, 2022, doi: 10.32604/iasc.2022.022241.
[5] C. Feichtenhofer, H. Fan, J. Malik, and K. He, “SlowFast Networks for Video Recognition,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, Oct. 2019, pp. 6201–6210. doi: 10.1109/ICCV.2019.00630.
[6] L. Shi, Y. Zhang, J. Cheng, and H. Lu, “Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA: IEEE, Jun. 2019, pp. 12018–12027. doi: 10.1109/CVPR.2019.01230.
[7] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, “NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2684–2701, Oct. 2020, doi: 10.1109/TPAMI.2019.2916873.
[8] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” arXiv preprint arXiv:1804.02767, Apr. 2018.
[9] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” arXiv preprint arXiv:2004.10934, Apr. 2020.
[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, and C.-Y. Fu, “SSD: Single Shot MultiBox Detector,” in ECCV, 2016, pp. 21–37.
[11] K. Simonyan and A. Zisserman, “Two-Stream Convolutional Networks for Action Recognition in Videos,” in NIPS, 2014, pp. 568–576.
[12] L. Wang, W. Li, R. Hu, and Z. Wang, “Violence Detection in Video Using Spatiotemporal Convolutional Neural Networks,” IEEE Trans. Circuits Syst. Video Technol., vol. 29, no. 3, pp. 666–679, Mar. 2019.
[13] X. Zhang, F. Liu, X. Shen, J. Liu, and Y. Tian, “Flow-Guided Feature Aggregation for Video Object Detection,” in ICCV, 2019, pp. 2701–2710.
[14] Z. Zhang, Y. Qi, J. Wang, G. Cheng, and X. Wang, “Deep Video Anomaly Detection: A Survey,” IEEE Access, vol. 8, pp. 129901–129917, 2020.
[15] M. Fayyaz, M. F. Hossain, and H. M. Rahman, “Real-Time Crowd Violence Detection Using YOLOv4 and LSTM,” Multimedia Tools Appl., vol. 80, pp. 15227–15245, 2021.
[16] S. Saha, S. Benenson, M. Fritz, and B. Schiele, “Deep Learning for Detecting Violence in Videos,” in ICCV Workshops, 2017, pp. 1–9.

[17] S. Herath, M. Harandi, and F. Porikli, “Going Deeper into Action Recognition: A Survey,” Image Vis. Comput., vol. 60, pp. 4–21, Feb. 2017.
[18] S. Singh, B. S. Manjunath, and M. Haque, “Spatiotemporal Semantics for Video Surveillance,” IEEE Trans. Inf. Forensics Security, vol. 15, pp. 2479–2494, 2020.
[19] G. Wang and L. Zhang, “End-to-End Temporal Instance Segmentation for Video Anomaly Detection,” in CVPR, 2021, pp. 15480–15489.
[20] M. Li, J. Chen, H. Wang, and R. Chellappa, “Deep-Learning-Based Video Anomaly Detection: A Review,” ACM Comput. Surv., vol. 54, no. 5, pp. 1–38, Sep. 2022.

POSTER

Faculty of Information and Communication Technology

HOME SURVEILLANCE IN GENERAL

Gue Kai Kit

Supervisor: Prof. Dr. Leung Kar Hang

01

INTRODUCTION

The growing concerns about home intrusion and unauthorized access have increased the demand for effective home security. An advanced home surveillance systems offer a crucial solution to enhance protection and ensure homeowners' peace of mind.

OBJECTIVE

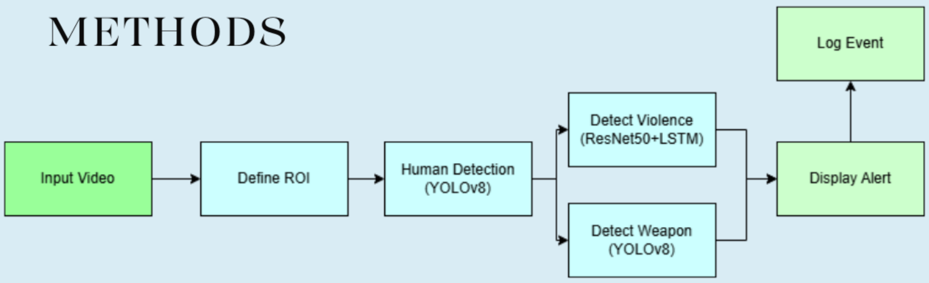
To develop an advanced home surveillance system that:

1. Mitigates the need for high levels of human supervision
2. Addresses the high false alarm rates
3. Implement a robust real-time alert system

02

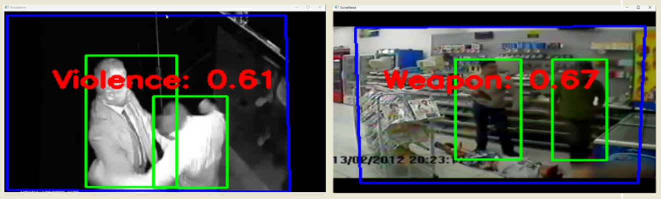
03

METHODS



RESULTS

Test result of the system to detect home intrusion



04