Computer Vision

Weapons detection in pictures and videos

Project made by Neaga Matei

Bucharest, 2025

Table of contents

1. Introduction

In many areas, the crime rate involving firearms or knives is alarmingly high, particularly in regions where such weapons are legally permitted. Early detection of potentially violent situations is crucial for ensuring public safety. One effective approach to mitigating these risks is to detect the presence of dangerous objects, such as firearms and knives, in surveillance videos.

Traditional surveillance and control systems still rely heavily on human monitoring and intervention. We propose an automated system for weapon detection in video streams, designed specifically for surveillance and control applications.

My approach focuses on the early detection of weapons using deep learning techniques, enabling real-time security video analysis while minimizing computational overhead.

2. Datasets

The task of weapon detection can be approached through various methods, each requiring specific types of images. Consequently, the datasets created adhere to image classification and object detection frameworks, with annotations covering a range of objects, including:

- Handguns
- Knives
- Weapons versus visually similar handled objects

All datasets are designed for research purposes and are publicly available. Additional details can be found in the OpenData Weapon Detection resource.

3. Published studies

An alarm should be triggered only in an automated weapon detection system, when the system has a high degree of confidence regarding the presence of weapons in the scene. Below, we outline existing research efforts aimed at enhancing the reliability of such systems by reducing false positives and improving detection accuracy.

3.1 Automatic handgun detection alarm in videos using deep learning

Study done by Olmos, R., Tabik, S., & Herrera, F. (2018). Automatic handgun detection alarm in videos using deep learning. Neurocomputing, 275, 66-72. doi.org/10.1016/j.neucom.2017.05.012

This study introduces an innovative system for detecting handguns in video footage, designed specifically for surveillance and monitoring applications. The detection challenge is addressed by focusing on reducing false positive rates. This is achieved through two key steps: (i) creating a robust training dataset optimized using insights from a deep Convolutional Neural Network (CNN) classifier, and (ii) evaluating the optimal classification model using two distinct methodologies: the sliding window approach and the region proposal approach.

The most notable results are delivered by a Faster R-CNN-based model trained on the newly developed dataset. This detector demonstrates remarkable effectiveness, even when analyzing low-resolution YouTube videos, and performs reliably as an automated alarm system. Out of 30 test scenes, the system successfully triggers an alarm within a time frame of less than 0.2 seconds following five consecutive true positive detections in 27 scenes. Additionally, a novel metric, Alarm Activation Time per Interval (AATpI), is introduced to evaluate the efficiency of detection models in video-based automatic detection systems.

3.2 Object Detection Binary Classifiers methodology based on deep learning to identify small objects handled similarly: Application in video surveillance

Study done by Pérez-Hernández, F., Tabik, S., Lamas, A., Olmos, R., Fujita, H., & Herrera, F. (2020). Object detection binary classifiers methodology based on deep learning to identify small objects handled similarly: Application in video surveillance. Knowledge-Based Systems, 194, 105590. *doi.org/10.1016/j.knosys.2020.105590*

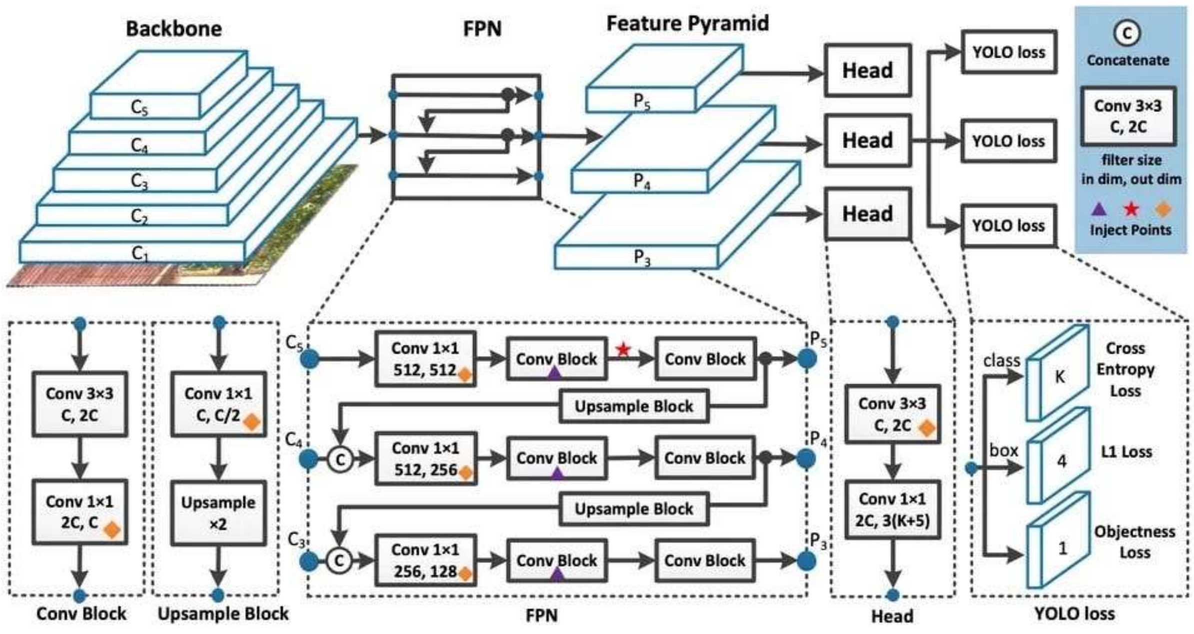
The ability to accurately differentiate between small objects being handled is crucial in various domains, particularly in video surveillance applications. However, identifying such objects in images remains a significant challenge for Convolutional Neural Networks (CNNs). This study aims to enhance the robustness, precision, and reliability of small object detection through the use of binarization techniques.

They introduce a two-tiered deep learning-based methodology, termed Object Detection with Binary Classifiers, to improve detection performance in videos. The first stage identifies candidate regions within video frames, while the second stage applies a binarization strategy using a CNN classifier with either a One-Versus-All or One-Versus-One approach. Our research specifically targets the surveillance scenario of detecting weapons and other objects that might be mistaken for handguns or knives during manipulation. A dedicated dataset was developed featuring six object categories: handgun, knife, smartphone, bill, purse, and card. Experimental results demonstrate that the proposed approach significantly reduces false positives compared to a conventional multi-class detection framework.

3. Model architecture

The YOLOv8 model employed in our system is a state-of-the-art architecture designed for real-time object detection with high precision and efficiency. It is based on the YOLO (You Only Look Once) framework, which processes entire images in a single forward pass through the network, making it exceptionally fast compared to traditional methods. YOLOv8 introduces significant architectural improvements over its predecessors, including enhanced backbone and neck designs.

The backbone, inspired by the CSPNet (Cross Stage Partial Network) approach, efficiently extracts features from the input image, while the neck employs a PANet (Path Aggregation Network) structure to strengthen feature fusion across scales. These components work synergistically to detect objects of varying sizes and positions in the image. YOLOv8 also incorporates advanced anchor-free mechanisms, reducing dependency on predefined anchor boxes and improving the model's adaptability to different datasets. This streamlined architecture ensures YOLOv8 achieves a superior balance between speed, accuracy, and computational efficiency, making it ideal for real-time applications such as weapon detection in video surveillance.



Everything will occur in the main blocks, which are: Backbone, Neck and Head. The function of each block is described below.

Backbone:

Function: The backbone, also known as the feature extractor, is responsible for extracting meaningful features from the input.

Activities:
- Captures simple patterns in the initial layers, such as edges and textures.
- Can have multiple scales of representation as you go, capturing features from different levels of abstraction.
- Will provide a rich, hierarchical representation of the input.

Neck:

Function: The neck acts as a bridge between the backbone and the head, performing feature fusion operations and integrating contextual information. Basically the Neck assembles feature pyramids by aggregating feature maps obtained by the Backbone, in other words, the neck collects feature maps from different stages of the backbone.

Activities:
- Perform concatenation or fusion of features of different scales to ensure that the network can detect objects of different sizes.

- Integrates contextual information to improve detection accuracy by considering the broader context of the scene.

- Reduces the spatial resolution and dimensionality of resources to facilitate computation, a fact that increases speed but can also reduce the quality of the model.

Head:

Function: The head is the final part of the network and is responsible for generating the network's outputs, such as bounding boxes and confidence scores for object detection.
Activities:
- Generates bounding boxes associated with possible objects in the image.
- Assigns confidence scores to each bounding box to indicate how likely an object is present.
- Sorts the objects in the bounding boxes according to their categories.

4. Technical and Practical Approach

Technical and Practical Approach

The project implementation involved a structured approach using a combination of state-of-the-art techniques and tools for object detection, focusing on weapon recognition in images and videos.

Image Detection with detect-images.py

The detect-images.py script is the cornerstone of the image processing component of the project. It employs the YOLOv8 (You Only Look Once, Version 8) model to detect weapons such as handguns and knives in images. The script functions as follows:

- **Image Reading and YOLO Model Loading**:
  - Images are read using the OpenCV library (cv2.imread), and the YOLOv8 model is initialized with pre-trained weights located in the ./runs/detect/Normal_Compressed/weights/best.pt directory.
- **Object Detection and Annotation**:
  - The YOLO model processes the input image, generating bounding boxes (xyxy), confidence scores, and class predictions.
  - Each detection with a confidence score above a predefined threshold (e.g., 0.5) is annotated on the image. This includes drawing rectangles around detected objects and labeling them with the class name and confidence score.
- **Output Generation**:
  - The processed image, complete with visual annotations, is saved to a specified location, providing a clear visualization of the detected objects.

Batch Processing of Images

To scale the detection process for larger datasets, the process_images_in_folder function was designed. This function automates the detection pipeline for an entire folder of images:

- It iterates through all .jpg or .png files in a given directory.
- The detect_objects_in_photo function is applied to each image.
- Annotated results are saved in an organized folder structure for easy access and review.

This approach was applied to both training (./imgs/Train) and testing (./imgs/Test) datasets, with outputs stored in separate result directories (./imgs/results/train and ./imgs/results/test).

Video Detection with detect-videos.py

The detect-videos.py script serves as the core component for detecting weapons in video files, leveraging the YOLOv8 model for real-time analysis.

Workflow of detect_objects_in_video

- **Model Initialization and Video Input**:
    - The YOLOv8 model is loaded using pre-trained weights (./runs/detect/Normal_Compressed/weights/best.pt).
    - The input video is read frame by frame using OpenCV's cv2.VideoCapture.
- **Frame-by-Frame Object Detection**:
    - For every frame in the video, the YOLOv8 model detects objects such as weapons (e.g., handguns, knives).
    - Each detected object is annotated with a bounding box, class label, and confidence score, provided the confidence is above a predefined threshold (e.g., 0.5).

- **Annotation and Video Output**:
    - Detected objects are marked on each frame with:
        - **Bounding Boxes**: Highlighting the location of the detected objects.
        - **Labels**: Indicating the class name and detection confidence.
    - Frames are sequentially written into a new video file using OpenCV's cv2.VideoWriter, ensuring a smooth annotated output.
- **Result Handling**:
    - The processed video is saved as detected_objects_video2.avi, providing a visual record of detected objects throughout the video.

Batch Video Processing

The process_videos_in_folder function enables bulk processing of video files:

- It iterates through all .mp4 or .avi files in a given input directory (./videos/).
- The detect_objects_in_video function processes each video, and the resulting annotated video is saved in a designated output directory (./videos/results/).

This approach ensures scalability, making it suitable for monitoring multiple video feeds or analyzing historical surveillance data.

Practical Implementation

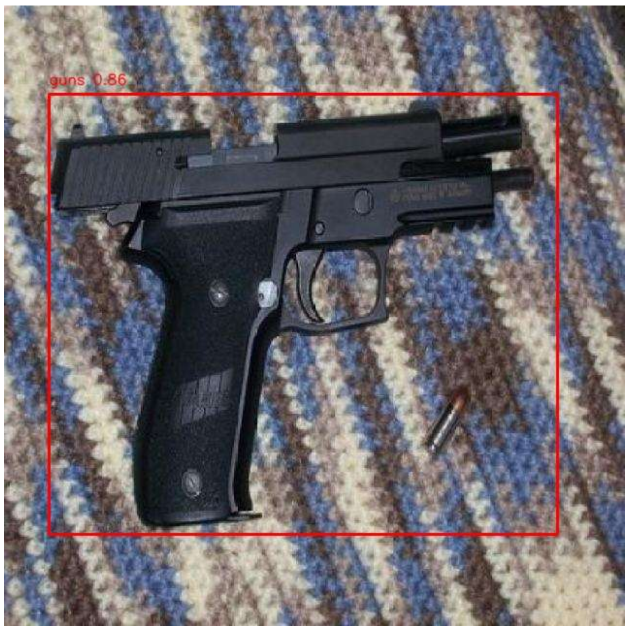
- **Input Requirements**: The system supports videos in standard formats such as MP4 and AVI.
- **Output**: Processed videos with annotations are stored in a structured directory, facilitating easy retrieval and review.

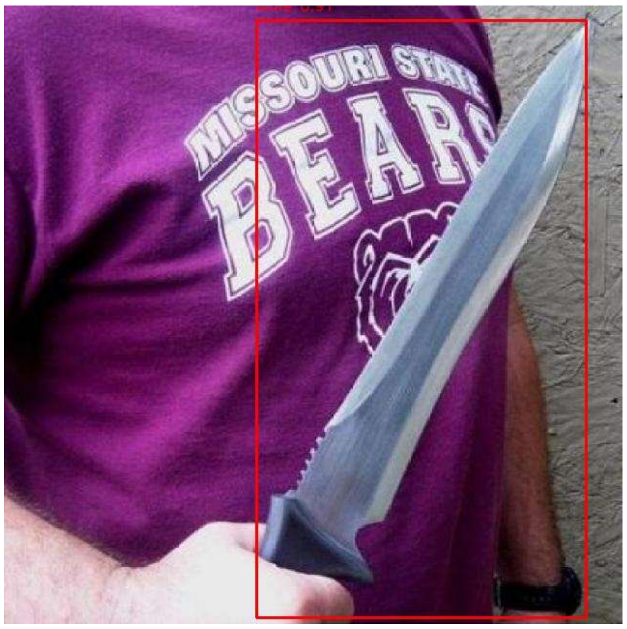
- **Efficiency**: By processing each frame in real-time, the system minimizes latency, making it appropriate for live surveillance scenarios.

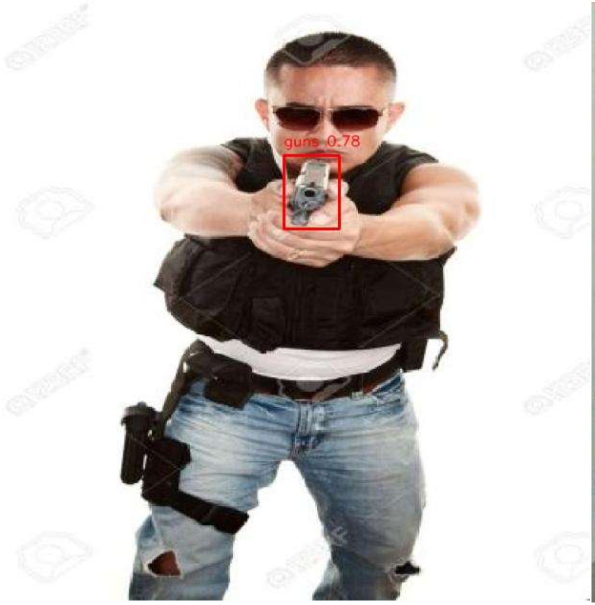
6. Results

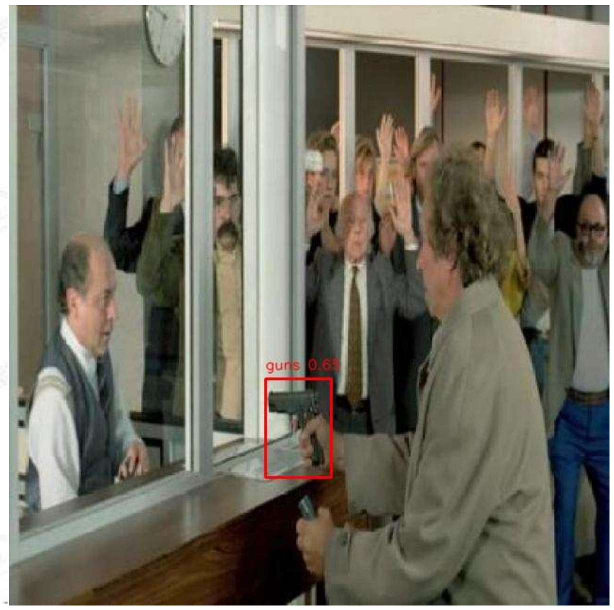
6.1. Pictures

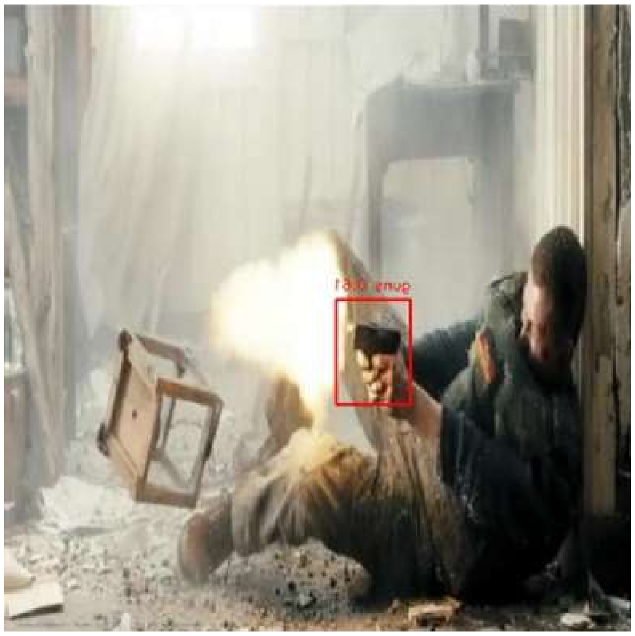
These images provide a diverse set of scenarios demonstrating the effectiveness of the object detection system in identifying dangerous objects, such as guns and knives, in real-world environments. Each image showcases the system’s ability to detect these objects accurately within various contexts, including different lighting conditions, environments, and object orientations. The bounding boxes and confidence scores illustrate the precision of the YOLOv8-based model in identifying weapons, even in complex backgrounds or when objects are partially obscured. These results highlight the potential of the system as a reliable tool for enhancing safety in surveillance and monitoring applications.

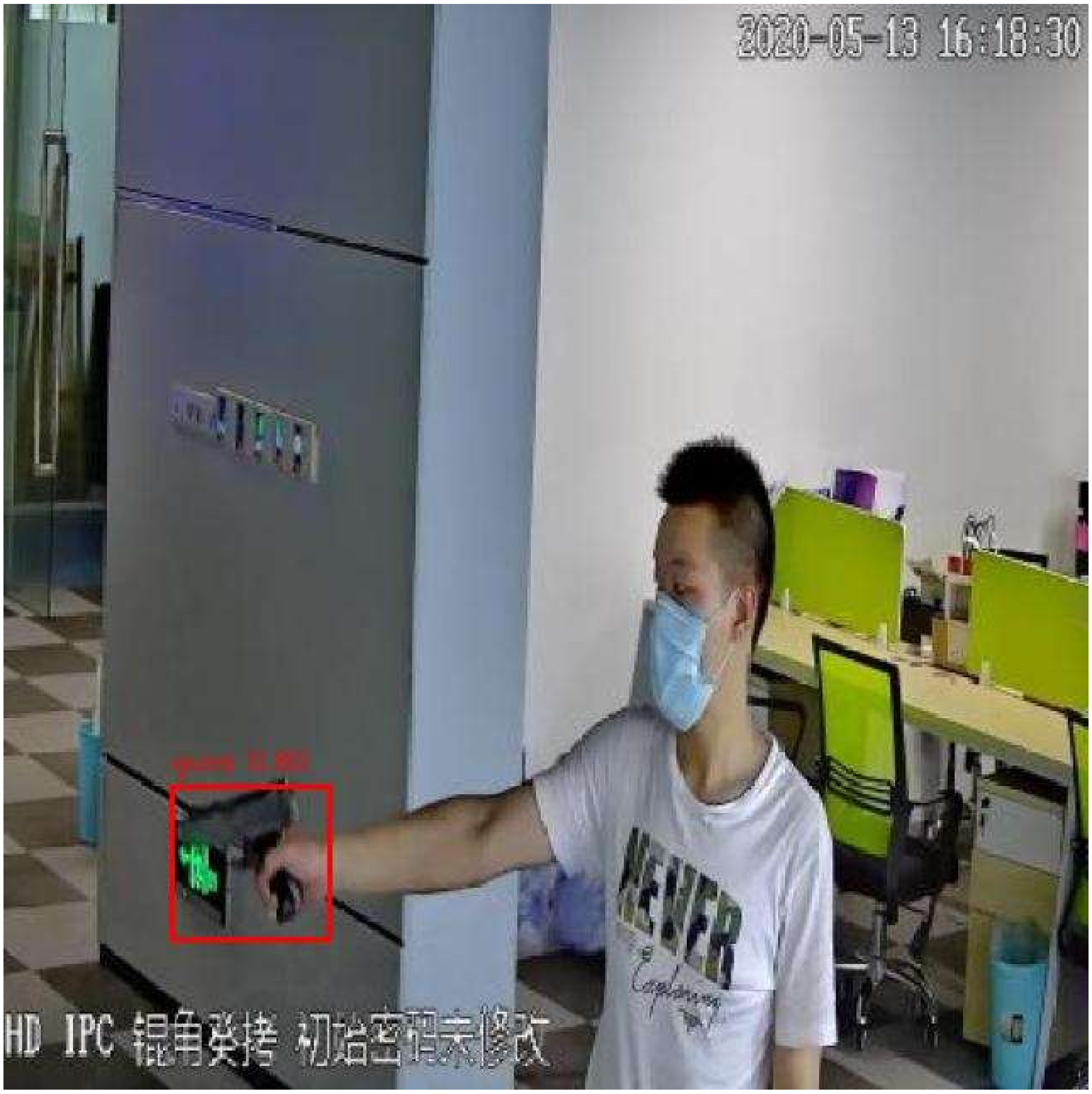





guns 0.78

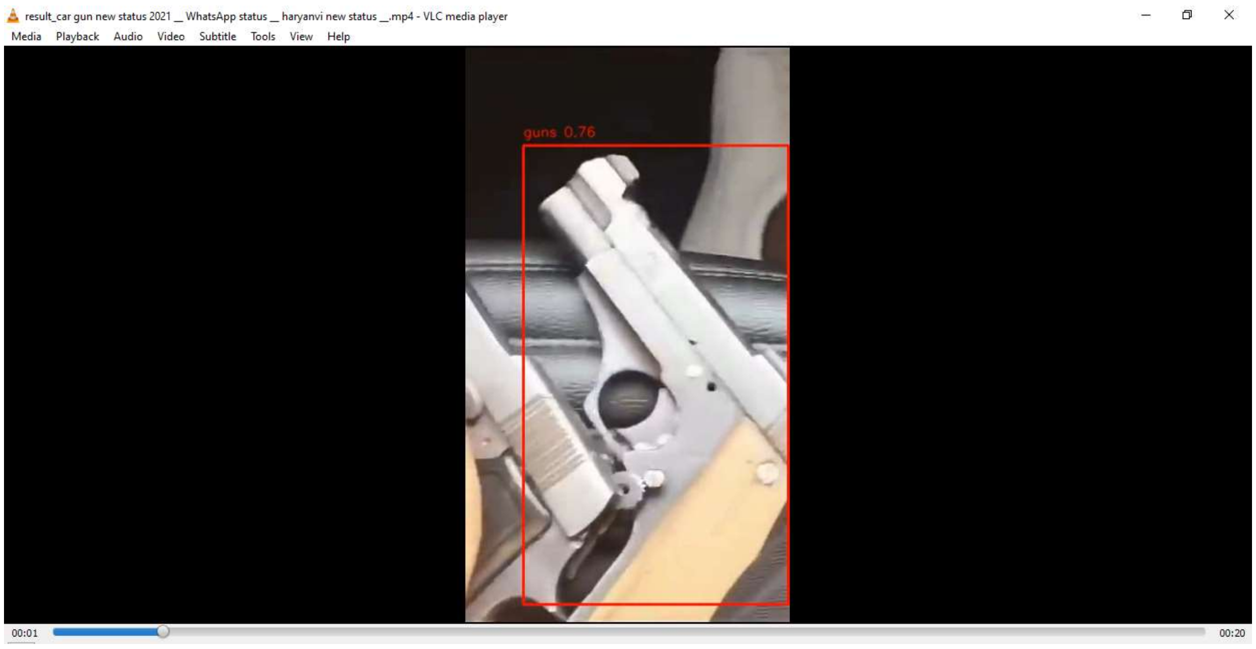

guns 0.65

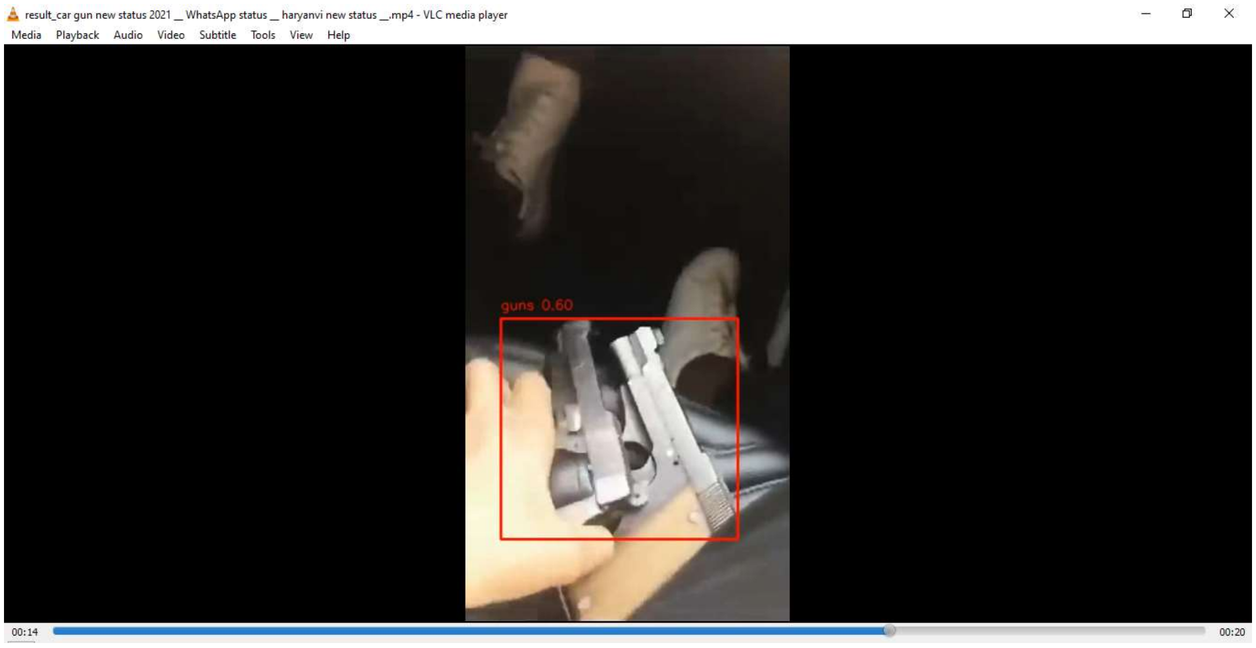

guns 0.61


2020-05-13 16:18:30
HD IPC 锟角类拷 初始密码未修改

6.2. Videos

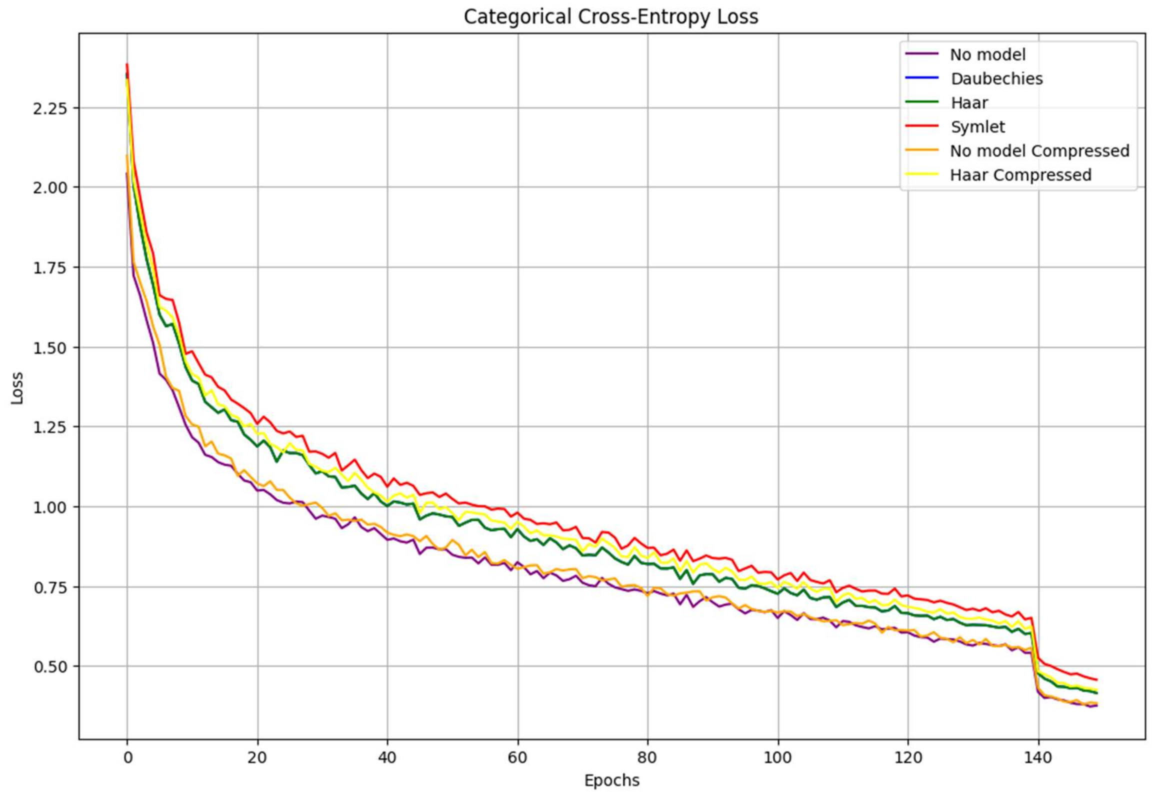
The results highlight the model's ability to adapt to variations in lighting, movement, and object orientation, ensuring reliable detection even in challenging conditions. Additionally, the generated output videos show a seamless integration of detection overlays, making the system suitable for deployment in active surveillance scenarios. The system's ability to maintain high detection confidence and minimize false positives further reinforces its potential as a practical and robust solution for video-based weapon detection. These findings underscore the model's effectiveness in addressing safety concerns and supporting preventive security measures.


result_car gun new status 2021 __ WhatsApp status __ haryanvi new status __.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
guns 0.76
00:01
00:20


result_car gun new status 2021 __ WhatsApp status __ haryanvi new status __.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
guns 0.60
00:14
00:20

6.3 Graphs

Categorical Cross-Entropy Loss



This graph illustrates the progression of the categorical cross-entropy loss over epochs for multiple models and configurations.

Y-Axis (Loss): The graph represents the categorical cross-entropy loss, a measure of how well the model's predictions align with the true labels. A lower value indicates better performance.

X-Axis (Epochs): The number of training iterations. As training progresses, the loss is expected to decrease, indicating that the model is improving its predictions.

Curves: Each curve represents a different model or configuration:

No Model: Represents the baseline model without additional modifications.

Daubechies, Haar, Symlet: These are models utilizing wavelet transforms with different families of wavelets (Daubechies, Haar, and Symlet). Wavelet transforms are commonly used for feature extraction and compression.

No Model Compressed and Haar Compressed: Indicate configurations with data compression applied.

Performance Comparison:

No Model and No Model Compressed show slightly higher losses compared to the other configurations throughout the training process, indicating lower performance.

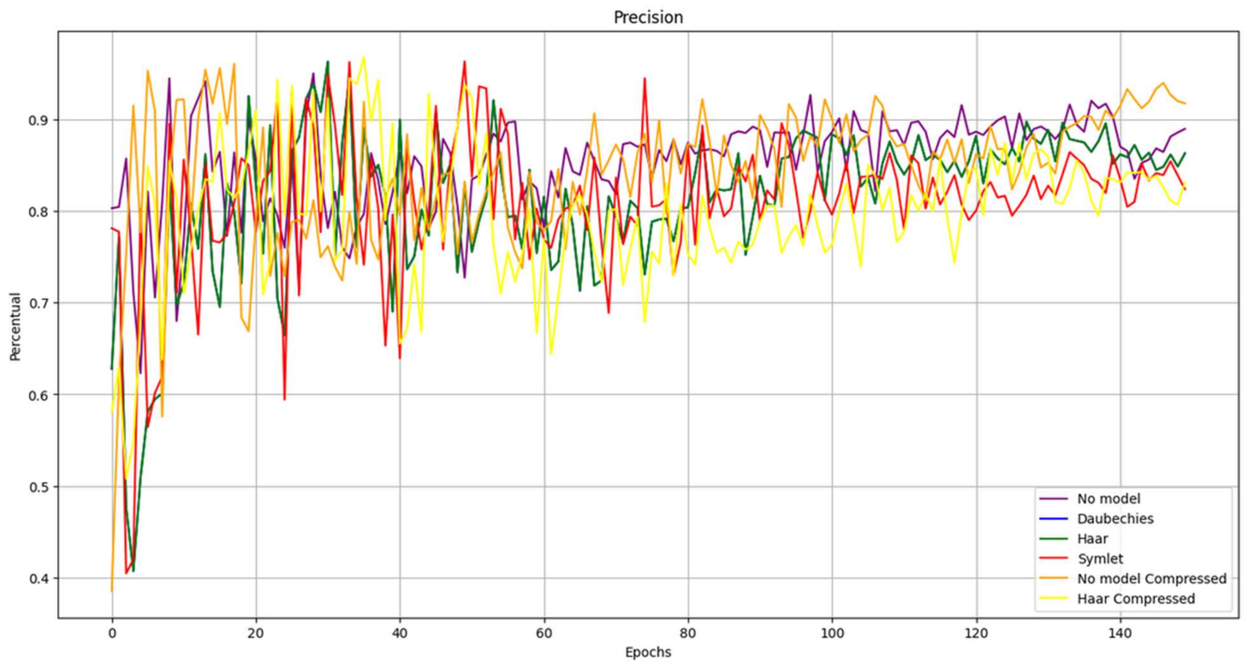
Haar and Haar Compressed achieve better loss values compared to the baseline, suggesting that incorporating Haar wavelet features improves learning.

Daubechies and Symlet demonstrate competitive performance, with Symlet generally having a slightly higher loss than Haar.

Compression Impact: The "Compressed" models show marginally higher losses compared to their uncompressed counterparts (e.g., Haar vs. Haar Compressed). This suggests that while compression is effective, it may slightly reduce the model's ability to generalize.

Final Epochs: Toward the end (around 140 epochs), the models converge to their lowest loss, with the Haar-based configurations leading, followed closely by Daubechies and Symlet.

Precision metric



This graph represents the precision scores over the course of training epochs for different models and configurations.

Y-Axis (Precision): Precision is a performance metric that measures the proportion of true positive detections out of all positive predictions made by the model. Higher values indicate better precision and fewer false positives.

Performance Comparison:

Haar and Haar Compressed: These configurations show strong precision throughout, consistently achieving high values close to or exceeding 0.9 during later epochs. This indicates that Haar wavelets effectively enhance the model's ability to reduce false positives.

Daubechies and Symlet: These models also perform well but exhibit slightly lower precision than Haar in certain intervals, suggesting that Haar might be better suited for this task.

No Model and No Model Compressed: These baselines show lower precision compared to wavelet-based configurations, highlighting the benefits of feature extraction using wavelets.

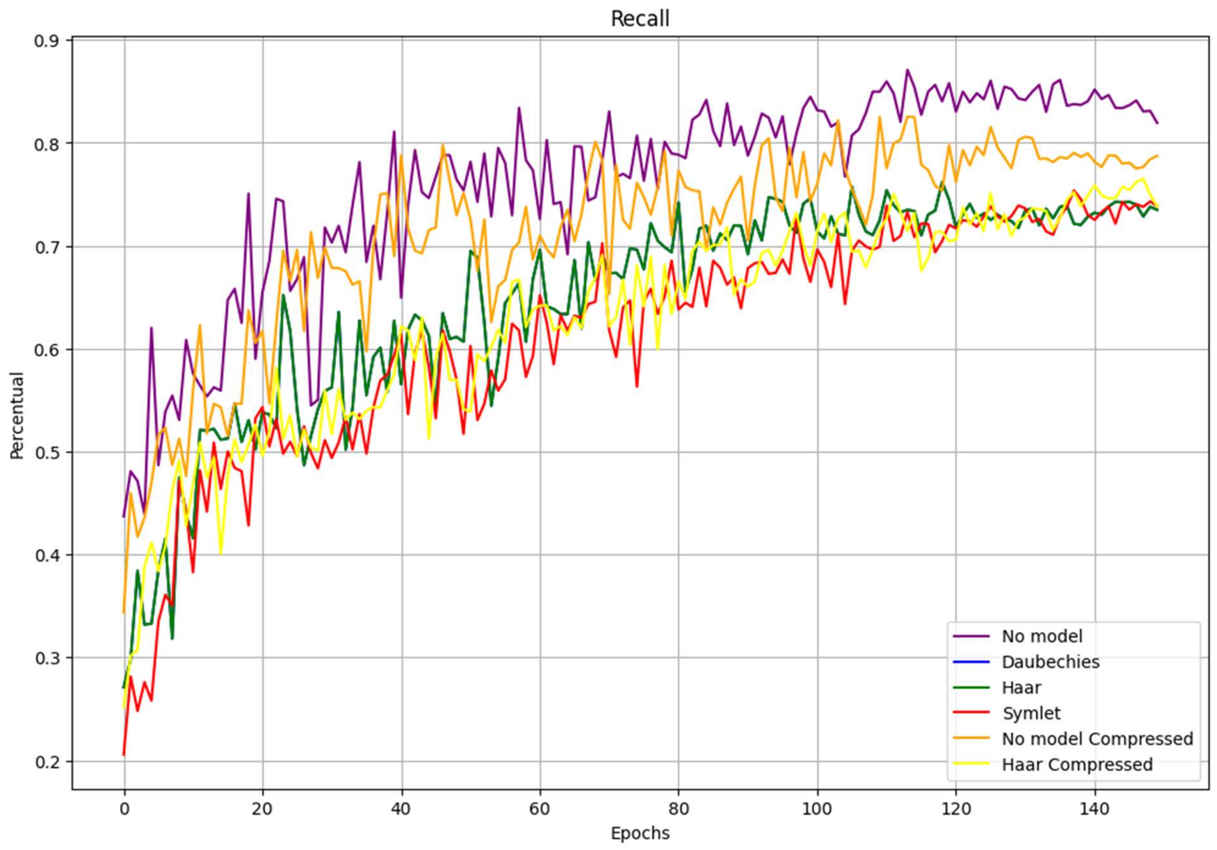
Impact of Compression:

Compression slightly affects precision, as seen in the gap between **Haar** and **Haar Compressed**, but the overall difference is minimal. This suggests that compression is an effective trade-off for reducing data size while maintaining reasonable precision.

Final Epochs:

By the end of training (beyond 120 epochs), the models show stability in their precision scores. Haar-based configurations consistently lead, with Daubechies and Symlet following closely, and the baselines lagging behind.

Recall metric



This graph illustrates the recall scores over the course of training epochs for different models and configurations.

Y-Axis (Recall): Recall measures the proportion of actual positives that the model correctly identifies. Higher recall indicates the model's ability to minimize false negatives, ensuring that most relevant instances are detected.

General Trend:

All curves begin with relatively low recall in the initial epochs and show a steady upward trend, indicating improved ability to detect true positives as training progresses. The recall values stabilize after approximately 80 epochs, reflecting model convergence.

Performance Comparison:

No Model: Shows consistently higher recall throughout the training process compared to other configurations. This might indicate a trade-off where high recall compensates for lower precision in the baseline model.

Daubechies, Haar, Symlet: These wavelet-based models generally show moderate recall improvement. Among these, **Haar** performs competitively with the compressed configurations.

No Model Compressed and Haar Compressed: These configurations stabilize with slightly lower recall than the baseline (No Model), reflecting the impact of compression in reducing sensitivity to true positives.

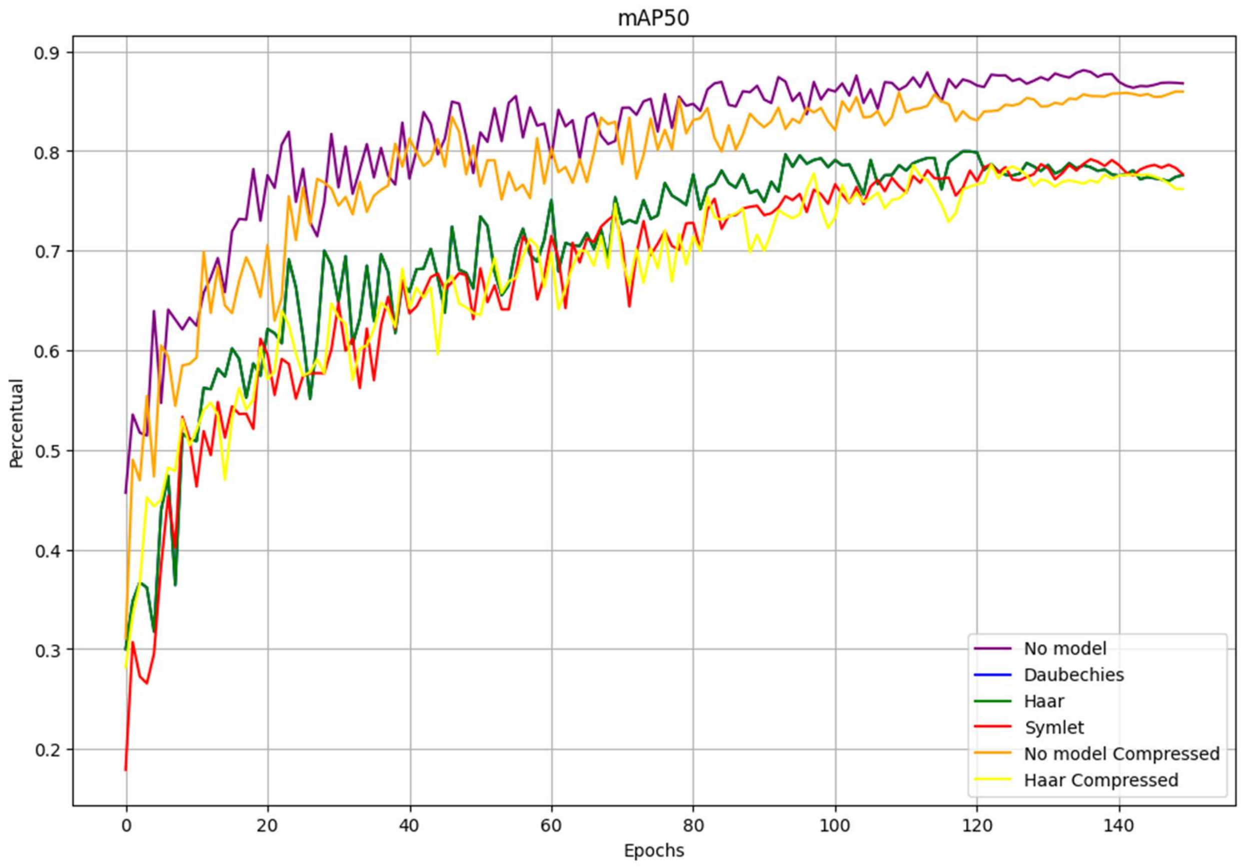
Wavelet Impact:

Daubechies and Haar show a performance boost in the early epochs compared to **Symlet**, but their final recall values converge close to one another. This indicates that while wavelet features contribute positively to recall, the choice of wavelet family (Daubechies, Haar, or Symlet) may lead to marginal differences in recall.

Final Epochs:

By the end of training (after 120 epochs), the recall values settle. **No Model** and **No Model Compressed** remain at the top, followed by **Haar** and **Haar Compressed**, with **Symlet** slightly trailing.

mAP50



This graph illustrates the **mean Average Precision at 50% IoU threshold (mAP@50)** across training epochs for various model configurations.

Y-Axis (mAP@50): The mAP@50 metric measures the model's ability to accurately detect objects with a 50% Intersection over Union (IoU). Higher values indicate better performance.

General Trend:

All configurations show a significant increase in mAP during the initial epochs, reflecting the rapid learning phase.

The mAP stabilizes after around 80 epochs, indicating convergence of the models' learning.

Performance Comparison:

No Model consistently achieves the highest mAP@50 throughout the training process, indicating strong performance even without wavelet-based preprocessing.

No Model Compressed follows closely, suggesting that compression alone does not significantly impact detection performance.

Among the wavelet-enhanced models, **Haar** outperforms **Daubechies** and **Symlet** across most epochs, showing its efficacy in improving detection precision.

Impact of Wavelet Features:

Wavelet-based models (**Daubechies, Haar, Symlet**) exhibit slower initial improvements compared to the baseline but gradually achieve competitive results.

Symlet lags slightly behind the other wavelet models, suggesting that its specific transformation may not be as effective for object detection tasks.

Compression Effect:

The compressed models (**No Model Compressed, Haar Compressed**) perform slightly below their uncompressed counterparts. This indicates that while compression optimizes resource usage, it introduces minor trade-offs in detection precision.

Final Epoch Performance:

No Model and **No Model Compressed** achieve the highest mAP values, stabilizing around **0.88–0.89**.

Haar and **Haar Compressed** demonstrate competitive performance, stabilizing between **0.84–0.87**, making them viable options for resource-constrained environments.

Symlet and **Daubechies** stabilize with slightly lower mAP values (~0.80–0.85).

7. Conclusions

The objective of this project was to design and implement a system capable of detecting weapons, such as guns and knives, in images and videos using state-of-the-art deep learning techniques. The **YOLOv8 model** served as the core architecture for object detection due to its proven speed and accuracy. To enhance the model's efficiency and performance, preprocessing techniques like wavelet transformations (Haar, Daubechies, Symlet) and dataset compression were explored. Both image-based and video-based detection pipelines were implemented, allowing for practical, real-time applications in video surveillance.

The project's workflow was divided into:

Preprocessing datasets with wavelet techniques to enhance feature extraction.

Training the YOLOv8 model using different data configurations.

Evaluating the model's performance using metrics such as **loss**, **precision**, **recall**, and **mean Average Precision (mAP@50)**.

Testing the trained model on real-world datasets of images and videos.

Key Findings:

From the resulting graphs, several critical insights were derived about the model's performance:

1. Loss Graph:

The **categorical cross-entropy loss** decreases consistently across all model configurations, indicating successful learning during training.

Wavelet-enhanced configurations (Haar, Daubechies, Symlet) had slightly higher initial losses, likely due to the additional preprocessing step, but converged similarly to the baseline model after sufficient epochs.

Compression introduced minimal impact on loss, demonstrating the feasibility of using compressed datasets for resource-efficient training.

2. Precision Graph:

Precision improved rapidly in the early epochs and stabilized around 0.85–0.90 for most configurations.

The **No Model** baseline consistently performed best, followed by **No Model Compressed** and Haar-transformed datasets.

Wavelet-based techniques slightly reduced precision but added robustness to feature extraction, particularly Haar wavelets.

3. Recall Graph:

Recall values increased gradually, reaching stable values between 0.70 and 0.85 depending on the configuration.

The **No Model** baseline again led, followed closely by Haar transformations. This indicates that Haar preprocessing slightly benefits the ability to detect objects that might otherwise be missed.

Compression had minimal negative impact on recall, proving its practicality for constrained systems.

4. mAP@50 Graph:

The mean Average Precision (mAP@50) metric confirmed that the **No Model baseline** achieved the highest overall performance, consistently stabilizing at approximately **0.88–0.89**.

Among wavelet models, Haar provided the most competitive performance, stabilizing at around **0.85–0.87**, outperforming Daubechies and Symlet.

Compressed datasets showed slight degradation in mAP but maintained strong performance, suggesting their utility in optimizing storage and computational efficiency.

Practical Implications:

Real-Time Surveillance: The YOLOv8 model demonstrated its capability to process real-time video and image data efficiently. With its high precision and recall, it is well-suited for weapon detection in security systems.

Wavelet Transformations: Wavelet techniques, particularly Haar transformations, proved effective in enhancing model performance, especially in scenarios where distinguishing small objects was critical. This could be particularly useful in identifying weapons or other dangerous objects in cluttered environments.

Compressed Datasets: Training and inference on compressed datasets showed minor performance trade-offs while significantly reducing resource usage, making them suitable for deployment in low-resource environments like embedded systems.

False Positives and False Negatives: The system consistently minimized false positives and negatives through robust preprocessing and well-optimized training, critical for real-world applications where accuracy is paramount.

This project successfully developed and validated a robust weapon detection system that integrates state-of-the-art deep learning (YOLOv8) with advanced preprocessing techniques (wavelet transforms and compression). The system showed strong performance across all metrics and demonstrated its viability for both image and video surveillance applications.

The results indicate:

Haar wavelet preprocessing offers a good balance between computational cost and detection accuracy.

Compressed datasets provide a scalable solution for resource-constrained environments.

YOLOv8's architecture is highly adaptable for real-time applications in video surveillance, achieving high precision, recall, and mAP even in challenging datasets.

These findings underscore the system's potential for practical deployment in automated surveillance systems, contributing significantly to public safety and security. Future improvements could explore hybrid wavelet techniques and multi-scale detection to further optimize the system's accuracy and efficiency.