Cutting Edge Weapon Detection in Real-Time CCTV videos

**Ajinkya Keshav Bamnolkar**   
*Dept. of Electronics and Tele-Communication Engineering  
G H Raisoni College of Engineering and Management*

Pune, India.  
ajinkyabamnolkar41@gmail.com

**Dr. Manisha Waje**  
*Dept. of Electronics and Tele-Communication Engineering  
G H Raisoni College of Engineering and Management* Pune, India.manisha.waje@raisoni.net**Ashutosh Rajesh Gadge**   
*Dept. of Electronics and Tele-Communication Engineering  
G H Raisoni College of Engineering and Management*

Pune, India.

aashutoshgadge07@gmail.com

**Kartik Ramhari Khakre**  
*Dept. of Electronics and Tele-Communication Engineering  
G H Raisoni College of Engineering and Managemen*t

Pune, India.  
kartikkhakre04@gmail.com

***Abstract—*****In today's world, ensuring public safety through advanced surveillance systems is critical. This paper presents a cutting-edge weapon detection system that operates in real-time using CCTV video streams, leveraging deep learning algorithms such as YOLO (You Only Look Once) and Detectron2. The system is designed to detect and classify weapons, such as guns and knives, with high accuracy while maintaining low latency. Through the use of optimized deep learning models, the system processes live video feeds and identifies potential threats in real-time, providing immediate alerts to security personnel.**

**The proposed system is scalable, supporting multiple video feeds. The paper explores the methodologies employed to achieve real-time processing, including frame extraction, model inference, and post-processing techniques like Non-Maximum Suppression. Optimization techniques such as quantization, pruning, and hardware acceleration are applied to ensure the system runs efficiently on edge devices. Extensive testing was conducted in real-world environments, demonstrating the system’s ability to handle diverse lighting conditions, occlusions, and varying weapon types. The results show significant improvements in detection accuracy, response time, and system reliability, making it a valuable addition to modern security solutions. Furthermore, the system adheres to privacy laws, ensuring compliance in surveillance applications. This paper contributes to the advancement of automated surveillance systems by providing an efficient and effective solution for weapon detection in public and private spaces.**

***Keywords: Weapon Detection, YOLO, Detectron2, Real-Time CCTV, Deep Learning, Security Systems.***

I. INTRODUCTION

In recent years, the need for enhanced security measures has become paramount in public and private spaces due to rising concerns over safety and crime. One critical advancement in this domain is the application of cutting-edge weapon detection technologies in real-time CCTV video surveillance systems. By leveraging sophisticated deep learning algorithms, such as YOLO (You Only Look Once), Faster R-CNN (Region-based Convolutional Neural Networks), Detectron2, and SSD (Single Shot MultiBox Detector), these systems can automatically identify and classify weapons within video streams as they are captured, facilitating timely interventions and improved situational awareness.

Traditional surveillance systems often rely on human operators to monitor video feeds, a method fraught with challenges, including fatigue and the potential for human error. In contrast, automated weapon detection systems offer a transformative approach by providing rapid, accurate detection capabilities. These systems not only enhance the speed at which threats can be identified but also reduce the likelihood of false positives and negatives, ensuring that security personnel are alerted only when necessary.

This paper presents a comprehensive examination of the methodologies, architecture, and performance evaluation of a real-time weapon detection system utilizing advanced deep learning techniques. We explore the integration of multiple object detection algorithms, highlighting their unique advantages and the synergistic benefits of their combined use in real-time applications. Our study aims to contribute valuable insights into the ongoing evolution of security technologies, demonstrating the potential of deep learning to revolutionize how we approach weapon detection in surveillance systems. Through rigorous testing and validation, we aim to illustrate the effectiveness of our proposed system in enhancing security protocols and safeguarding individuals in various environments.

CCTV cameras are essential for enhancing security and are widely used in public spaces for safety, crime investigation, and evidence collection. Countries like the UK, with approximately 4.5 million cameras, and China, boasting the world's largest surveillance system with 170 million cameras, exemplify the effectiveness of CCTV in crime reduction and rapid apprehension of suspects. For instance, Poland saw a 60% decrease in drug cases after installing just 450 cameras. However, traditional surveillance methods require constant human monitoring, which is often impractical due to attention fatigue when managing multiple screens.

To address this challenge, there is a need for advanced surveillance systems that can automatically detect weapons and alert security personnel. While previous studies primarily focused on concealed weapon detection using traditional machine learning, the advent of deep learning, particularly Convolutional Neural Networks (CNNs), offers promising advancements in automatic feature learning for improved object detection and categorization in real-time surveillance applications.[1]

This research focuses on optimizing video surveillance systems for the detection of armed individuals. By utilizing a widely adopted object detection algorithm from computer vision and deep learning, the study aims to identify people, faces, and handguns in video footage. The primary challenge is to accurately detect individuals carrying handguns while distinguishing their faces. The proposed solution seeks to automate the detection process, thereby reducing reaction times to crimes and enhancing the efficiency of security personnel's monitoring efforts.[2]

Pistols and revolvers are commonly used firearms in various crimes. According to a United Nations Office on Drugs and Crime (UNODC) report, which analyzed data from 81 countries, the most seized firearms due to illicit use include pistols (39%), shotguns (25%), rifles (18%), revolvers (14%), submachine guns (3%), and machine guns (1%). In the Americas, the homicide rate among men aged 18 to 19 is notably high at 46 per 100,000 individuals, with firearms being significantly more involved in homicides compared to other regions. These statistics highlight the importance of focusing this research on handgun use.

The research emphasizes the efficiency and speed of the You Only Look Once (YOLO) model in detecting objects in urban environments, outperforming traditional methods such as R-CNN, which rely on multiple detection stages. YOLO processes images in real-time with a single network pass, and its improved versions, YOLOv3 and YOLOv4, enhance precision, reduce false positives, and improve generalization, making them suitable for urban surveillance and traffic management.

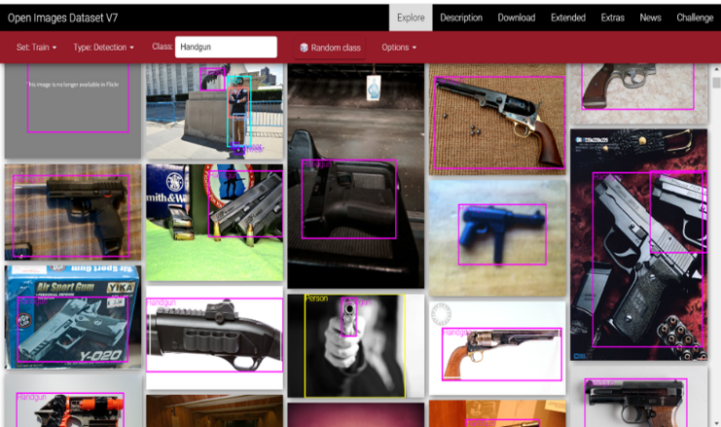
This study explores YOLO's adaptability to urban settings, presenting new opportunities for intelligent surveillance systems that enhance weapon detection. The findings have practical applications across public safety, traffic management, and industrial automation, significantly impacting urban safety and efficiency. Additionally, the results contribute to advancements in computer vision and deep learning by effectively managing multiple objects in real-time under changing urban conditions.

The article is structured into sections, starting with an introduction to the challenges of real-time object detection in urban areas, followed by "Materials and Methods," which outlines data collection, preprocessing, dataset selection and annotation, model architecture, training, and evaluation. The "Results" section compares the proposed model with others in terms of precision and efficiency, while the "Discussion" explores the implications of these findings and potential improvements. Finally, the "Conclusions" summarize the main achievements of the research.[3]

II. CUTTING EDGE WEAPON DETECTION IN REAL-TIME CCTV VIDEOS

1. *DATA COLLECTION*

Open Images V7 is a large-scale dataset for machine learning, primarily used for image recognition and object detection tasks. It contains millions of annotated images with rich labels, covering a broad range of objects and scenes. We have taken 3543 images of all classes inclusive Using the official Open Images Dataset downloader, we downloaded the images of classes

Handgun (1049), Rifle (1499), Knife (567), Axe (156) Shotgun (144), Sword (128).

1. *DATA ANNOTATION*

CVAT (Computer Vision Annotation Tool) is an open-source web-based tool designed to facilitate the annotation of image and video data for computer vision tasks. It provides an intuitive interface for annotating various types of objects and scenes, making it an essential tool for machine learning and AI projects that require labelled datasets.

To annotate data using CVAT AI , here’s the process we followed:

1. Setting up CVAT:

We installed and set up CVAT on our local machine, or in some cases, we used the cloud version. Once it was ready, we logged into the CVAT interface to manage annotation tasks.

2. Creating an Annotation Task:

We uploaded the images we wanted to annotate, from Open Images V7 . While creating the task, we defined the object classes, such as handguns, rifles, knives, axes, shotguns, and swords. We also configured the annotation settings, like choosing whether to use bounding boxes, polygons, or points.

3. Annotating Objects:

We began by selecting the object class from the interface and manually annotated the objects by drawing bounding boxes around the relevant weapons.



4. Review and Refinement:

After finishing the annotations, we thoroughly reviewed the labels to ensure accuracy. CVAT allowed us to zoom in and adjust bounding boxes as needed. For some tasks, we used CVAT’s AI-assisted tools, which suggested auto-generated labels. We reviewed and corrected these suggestions to maintain high-quality annotations.

5. Exporting Annotations:

Once we were satisfied with the annotations, we exported them in the required format Yolo 1.1 for YOLO v8 model training, CVAT images 1.1 for detectron2 . Before moving on, we ensured the dataset was balanced and properly labelled.

1. *MODEL TRAINING*

**A. YOLOv8**

We uploaded the annotated images in yolo 1.1 format in our google drive form there we trained the model

We imported the YOLO class from the Ultralytics library, which provided the YOLO models for object detection, segmentation, and classification tasks.

We loaded a new model configuration using YOLO("yolov8n.yaml"), which was the YOLOv8-nano version. This model configuration defined the architecture and hyperparameters, allowing us to build the model from scratch

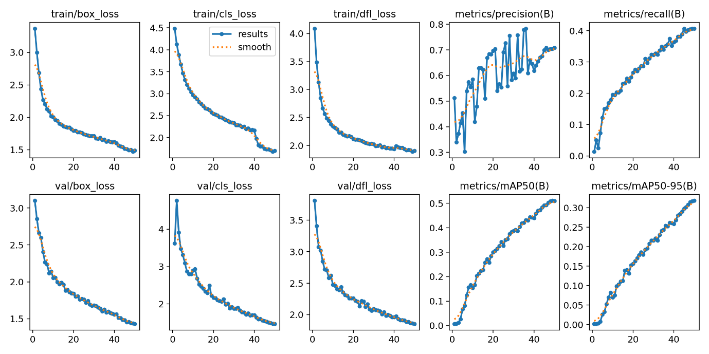
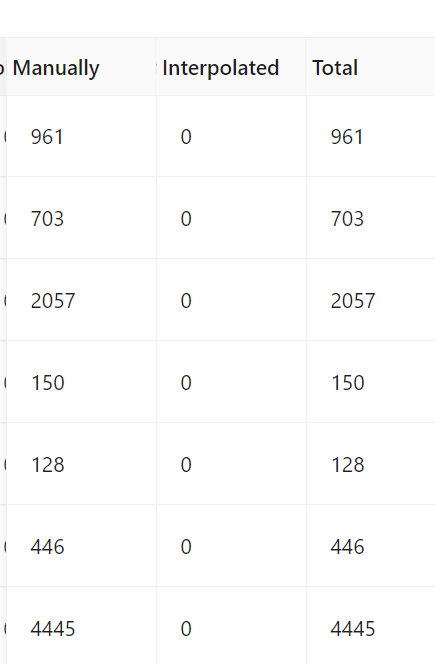
We initiated the training process by calling model.train() on the dataset we specified in the google\_colab\_config.yaml file.

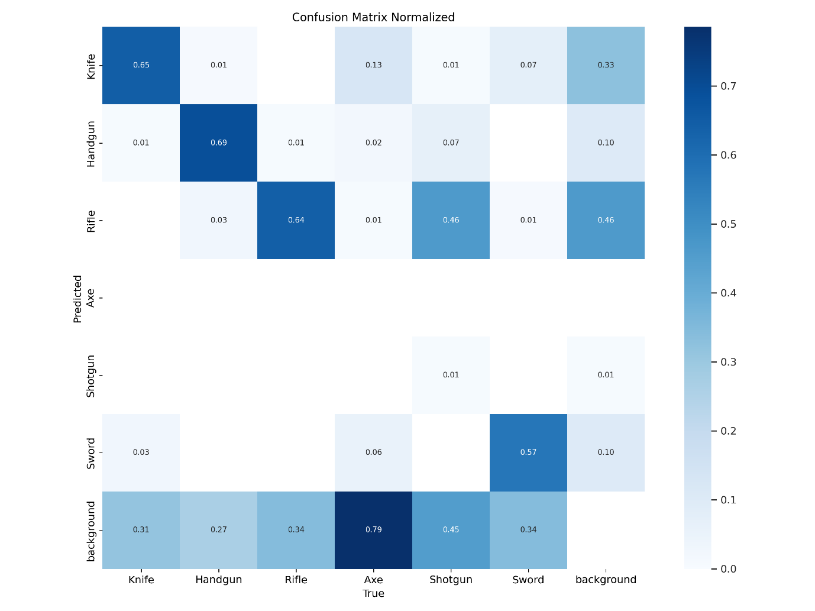
We passed the following parameters:

• data: We specified the dataset configuration using the path to the YAML file, which contained information about where our training and validation images were located, the number of classes, and the class labels.

• epochs=50: We set the training to run for 50 epochs, meaning the model processed the entire dataset 50 times to improve its predictions and accuracy.

Result after training the YoloV8 model:





**B. Detectron2**

Detectron2 [3] is a multi-purpose library from Facebook

that provides state-of-the-art object detection. It is based

on PyTorch and aimed to perform high-speed training. It is

written in a modular form to allow future work to be based

on this implementation. Detectron2 includes a number of object detection models like Faster R-CNN, DensePose, Cascade R-CNN, Mask R-CNN etc. It allows object detection and marking with bounding boxes, human pose estimation and segmentation masks. We used the default predictor, COCO-Instance Segmentation and Mask R-CNN with score threshold value (MODEL.ROI\_HEADS.SCORE\_THRESH\_

TEST) set to 0.5 for our experiments. A sample of Detectron2 features are shown in We used Detectron2 and YOLO, and showed that the approach is detector-agnostic and works

well with both combinations. We noticed that YOLO detects more meaningful objects compared Detectron2

**C. SSD**

As an object detection algorithm based on deep learning,

Single Shot MultiBox Detector (SSD) [8] has high performance in both detection accuracy and detection speed. SSD algorithm is proposed by Liu W et al. in 2016 to solve

the problem of insufficient detection accuracy of YOLO [9]

series algorithm in object positioning. Its main idea is to

sample densely and evenly at different locations of the image.

SSD draws on the concept of anchor in Faster R-CNN [10].

1. *MODEL SELECTION*

* Detectron2 is a comprehensive framework that offers a variety of models, including SSD, allowing for extensive customization and flexibility in detection tasks.
* SSD is a specific model focused on speed, suitable for real-time applications but with less emphasis on accuracy compared to more complex models.
* YOLOv8 is the latest in the YOLO series, optimized for speed and accuracy, making it an excellent choice for real-time object detection and related tasks.

We selected the yolov8 model for our Weapon Detection Next, we use Pycharm for using our Trained model for Real Time Weapon Detection:

First, we ensured that all the necessary libraries were installed. Since we were using YOLOv8 from the `ultralytics` package, we also needed `opencv-python` to handle real-time video input. So, we started by running:

“pip install ultralytics opencv-python”

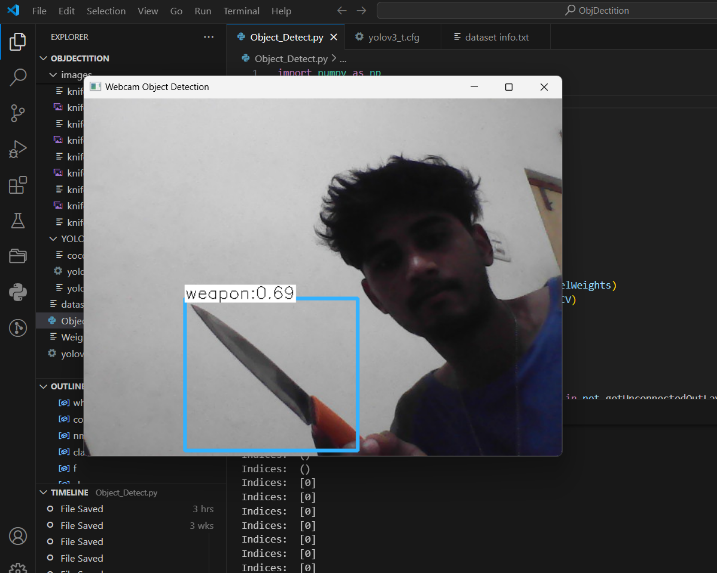
After we had the libraries set up, we proceeded to load the YOLOv8 model. We opted to use the `yolov8n.pt` model because it was lightweight and optimized for speed. However, if we had trained our own model specifically for weapon detection, we would have replaced `"yolov8n.pt"` with the path to our own model.

We also set a confidence threshold to 0.5 to filter out low-confidence detections, ensuring that only objects detected with higher certainty would be processed. This step was important to reduce false positives, especially since we were dealing with a critical task like weapon detection.

Next, we wrote a function to process each frame of our video stream. This function passed the frame through the YOLOv8 model, which returned detection results, including bounding boxes and confidence scores for any identified objects. For each object detected, we checked its confidence score, and if it was above our threshold, we labelled it as a potential weapon.

We then used the bounding box coordinates to draw rectangles around the detected weapons in the video feed. Additionally, we displayed the class name (gun, knife, Handgun Rifle, Knife, Axe, Shotgun, Sword) and the confidence score right above the bounding box, making it easy to identify the object visually.

Finally, we opened a connection to the webcam using `cv2.VideoCapture(0)` and continuously processed each frame in real-time, displaying the results with bounding boxes on the screen. If we wanted to stop the program, we simply pressed the 'q' key to break the loop and close the video feed.



1. *MULTIPLE CAMERA INPUT:*

We first identified the index for each camera (e.g., 0, 1, 2, etc.) depending on how our cameras were connected to the system. Each index refers to a different camera device. In the case of multiple cameras, we needed to open a separate VideoCapture stream for each one.

1. Opening Multiple Video Streams:
   * + We created two Video Capture objects: cap1 for the first camera and cap2 for the second one. The numbers 0 and 1 represent the camera indexes.
     + Each camera gets its own stream. If we wanted more cameras, we would simply add additional VideoCapture objects like cap3 = cv2.VideoCapture(2).
2. Reading Frames:

* In each iteration of the loop, we read frames from both cameras using cap1.read() and cap2.read(). This gives us two different frames, frame1 and frame2, which correspond to the first and second cameras.
* If reading from either camera fails (e.g., no frame is captured), we exit the loop to avoid errors.

1. Displaying the Feeds:

* We displayed each camera feed in separate windows using cv2.imshow(). This way, we could monitor both camera streams in real-time.
* Each window is labelled ('Camera 1', 'Camera 2'), making it clear which feed corresponds to which camera.

1. Stopping the Loop:

* To exit the loop and close the video feed, we pressed the 'q' key. Once this happens, we broke out of the loop and released both VideoCapture objects.

1. Releasing Resources:

* We made sure to release both cap1 and cap2 once we were done to free the camera resources. Then, we used cv2.destroyAllWindows() to close any OpenCV windows we had opened.

1. *TELEGRAM ALERT:*

**Step 1:** Create a Telegram Bot via BotFather.

We started by opening Telegram and searching for "BotFather." Once we were chatting with BotFather, we created a new bot by using the command /newbot. After setting a name and username for our bot, BotFather provided us with an API token, which we used to send alerts through our script.

**Step 2:** Set Up the Chat ID.

To send alerts to ourselves or a group, we needed the chat ID. We obtained this by either:

* Adding the bot to a Telegram group and using that group’s chat ID.
* If we wanted to receive the alerts personally, we could get our chat ID by messaging the bot first, then using a small script to capture it.

**Step 3**: Code to Send Alerts.

* Once we had the bot’s API token and the chat ID, we integrated the alerting functionality in our Python code. Whenever a specific event happened, like detecting a weapon, we triggered a Telegram alert.

**API Token and Chat ID**:

We obtained the API token from BotFather and assigned it to the variable bot\_token(@alert\_weapon\_bot). This token gave us access to the bot's API so we could send messages. The chat\_id is where we wanted to send the alerts. If it was a group chat, we got the group chat ID, or if it was a personal message, we used our own chat ID.

**Sending the Alert**:

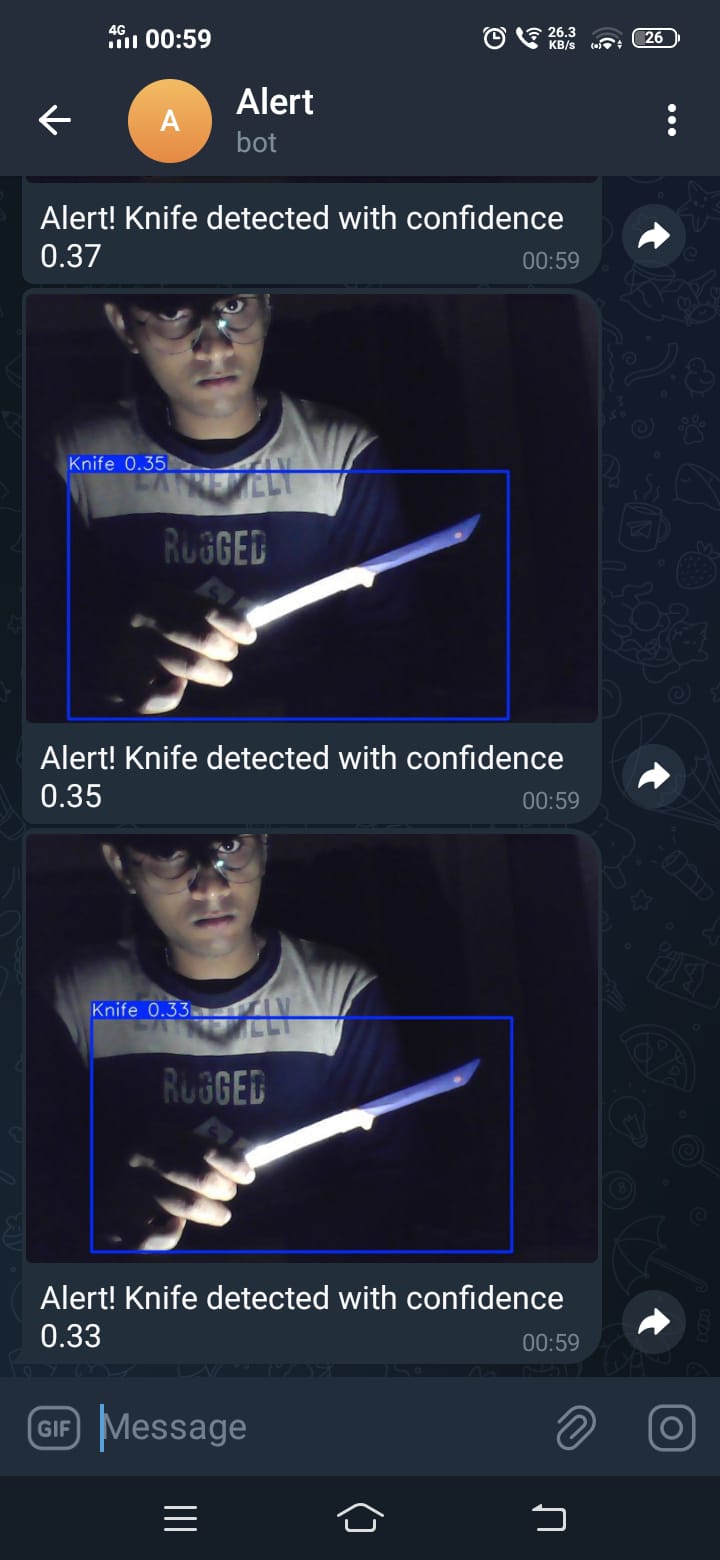
We used the Telegram Bot API endpoint to send messages. The URL https://api.telegram.org/bot{xxxxxxxxxx}/sendMessage was where we sent our requests. We replaced {bot\_token} with our actual token. The payload was a dictionary containing two key pieces of information: chat\_id (who we were sending the message to) and text (the actual message).

**Making the Request**:

To send the message, we used the requests.post() function, which made an HTTP POST request to Telegram’s API with our payload. If the request was successful, we printed a confirmation that the alert was sent. If not, we printed the error status code to debug.

**Triggering the Alert**:

Whenever a weapon was detected in our real-time system (for example, the variable weapon\_detected was set to True), we called the send\_telegram\_alert() function with a custom message. The message could be anything, such as “Weapon detected in the video feed!” or more detailed, depending on what we wanted to communicate.



1. *FUTURE WORK*

There are several areas where we can further enhance and optimize our weapon detection system. One of the key improvements lies in expanding the dataset. By training the YOLO model with a larger and more diverse set of images, particularly those depicting different types of weapons and environments (indoor, outdoor, various lighting conditions), we can significantly improve the accuracy and robustness of the model. Additionally, as newer versions of YOLO continue to be developed, we can integrate these more advanced models to achieve even better detection rates and faster inference times.

Another area for improvement is the notification system. Currently, we’ve implemented real-time alerts via Telegram, but expanding the system to support multiple alert channels would increase its flexibility and reach. Integrating notifications through platforms like WhatsApp, direct SMS via phone numbers, and email notifications would provide more options for timely alerts, ensuring that users are always informed in case of a detected threat, regardless of their preferred communication platform.

Furthermore, scaling the system for larger deployments, such as taking input from multiple cameras and CCTV systems, can be achieved by leveraging cloud services like AWS. Using AWS, we could deploy the model at scale, allowing it to process multiple video streams simultaneously. This would make the system suitable for larger environments, such as schools, shopping malls, or airports, where multiple points of surveillance are required.

1. *CONCLUSION*

In conclusion, our real-time weapon detection system using a pretrained YOLOv8 model has proven to be an effective and scalable solution for identifying potential threats. With its current capability of detecting weapons in real-time and sending alerts via Telegram, it already serves as a practical security tool. However, there is significant potential for future optimization and expansion. By training the model with more diverse datasets, adopting newer versions of YOLO, and incorporating additional alerting methods such as WhatsApp, SMS, and email, we can further improve both the accuracy and usability of the system. Additionally, integrating multiple camera feeds and leveraging cloud infrastructure like AWS would allow the system to be deployed in larger, more complex environments, making it highly adaptable for modern surveillance needs.

References

1. M. T. Bhatti, M. G. Khan, M. Aslam and M. J. Fiaz, "Weapon Detection in Real-Time CCTV Videos Using Deep Learning," in IEEE Access, vol. 9, pp. 34366-34382, 2021, doi:10.1109/ACCESS.2021.3059170.
2. M. Mynuddin, S. U. Khan, R. Ahmari, L. Landivar, M. N. Mahmoud and A. Homaifar, "Trojan Attack and Defense for Deep Learning-Based Navigation Systems of Unmanned Aerial Vehicles," in IEEE Access, vol. 12, pp. 89887-89907, 2024, doi: 10.1109/ACCESS.2024.3419800.
3. Van Vung Pham; Tommy Dang, Hands-On Computer Vision with Detectron2: Develop object

detection and segmentation models with a code and visualization approach, Packt Publishing, 2023.

1. A.J. Amado-Garfias, S. E. Conant-Pablos, J. C. Ortiz-Bayliss and H. Terashima-Marín, "Improving Armed People Detection on Video Surveillance Through Heuristics and Machine Learning Models," in IEEE Access, vol. 12, pp. 111818-111831, 2024, doi: 10.1109/ACCESS.2024.3442728.
2. J. Ruiz-Santaquiteria, A. Velasco-Mata, N. Vallez, G. Bueno, J. A. Álvarez-García and O. Deniz, "Handgun Detection Using Combined Human Pose and Weapon Appearance," in IEEE Access, vol. 9, pp. 123815-123826, 2021, doi: 10.1109/ACCESS.2021.3110335.
3. M. T. Bhatti, M. G. Khan, M. Aslam and M. J. Fiaz, "Weapon Detection in Real-Time CCTV Videos Using Deep Learning," in IEEE Access, vol. 9, pp. 34366-34382, 2021, doi: 10.1109/ACCESS.2021.3059170.

.