Journal Homepage: www. ijarpr.com ISSN: 3049-0103 (Online)

International Journal of Advance Research Publication and Reviews

Vol 02, Issue 10, pp 15-26, October 2025

Guard Vision: Realtime Weapon Detection

Prof. Sreeja. A. K.[1], Varsha A Habbu[2], Shamita M[3], Yashaswini S[4]

[1]Dept. of Computer Science BNM Institute of Technology, Bangalore, India sreejaak@bnmit.in
[2]Dept. of Computer Science BNM Institute of Technology Bangalore, India varshahabbu16@gmail.com
3Dept. of Computer Science BNM Institute of Technology Bangalore, India mshamitabhat@gmail.com
[4]Dept. of Computer Science BNM Institute of Technology Bangalore, India yashaswinisrinivas586@gmail.com

ABSTRACT—

Surveillance systems are fundamental in monitoring and ensuring public safety in various environments like public spaces and private residences. With increasing threats in everyday life, there is a requirement for better surveillance systems capable of accurate detection of weapons and other hazardous objects in such spaces. This paper implements a weapon detection system which is implemented using the YOLO algorithm and OpenCV library, that can be integrated with a CCTV surveillance to process images and videos and identify the weapons in the input provided. The combination of YOLO's deep-learning based detection and OpenCV's image manipulation abilities ensures that the system can detect weapons even under different external conditions. The system makes use of the Graphics Processing Units (GPUs) to accelerate the processing time taken by the model.

Keywords—Weapon detection, Deep learning, Artificial Intelligence, OpenCV, Image Classification, Video Surveillance, Machine learning, YOLO model

1. Introduction

Security threats has been a common issue from the past few decades and these threats are usually identified and processed by professionals like security staff and other law enforcement officers. This approach depended heavily on the expertise, experience and the interpretation of the threat by these professionals, which can be complex, time-consuming and prone to a lot of errors. The hunt to automate threat detection has been going on since the 21st century, with various achievements already made in the field. CCTV cameras are crucial in addressing this issue and are regarded as a key component of security systems. This paper implements real-time monitoring of weapons and its detection. A general weapon detection system analyzes visual data provided and then identifies and categorizes the potential data. This involves two key elements: the visual data provided and the predefined threat indicators

Aligning visual data with threat detection still remains a significant challenge in developing a robust, real-time system that can be classified into various types of weapons such as knives, guns, rifles and other hazardous objects in images and surveillance camera videos. By using Convolutional Neural Networks (CNNs) and other machine learning techniques, the system is created to accomplish high precision and reliability. This ensures accuracy in detection even in complex and dynamic environments.

The beginning of the development process is marked by the collection and preprocessing of various datasets, which include labelled images depicting different types of weapons. The collected data is vital for training and polishing the machine-learning model. By leveraging a pre-trained model of YOLO(You Only Look Once) on the advanced algorithm of CNN, the system reduces computational expenses and improves performance to make it more suitable for real-time applications.

Model effectiveness is another crucial factor to consider. Visualization tools like the confusion matrices and Receiver Operating Characteristic (ROC) curves are also used for analysing the performance and to identify areas where improvement is needed. The integration of transfer learning allows the system to quickly adapt to new datasets, helping in enhancing the versatility and efficiency.

The system is optimized for deployment in many real-life applications such as surveillance systems, security checkpoints, and autonomous monitoring solutions. Hardware acceleration, mainly through GPU utilization, is made use of to fulfil the many demands of detecting weapons for faster and more accurate response times.

2. LITERATURE SURVEY

Harsh Jain et al. (2020) proposed the implementation of automatic gun (or) weapon detection by using a convolution neural network (CNN) based SSD and Faster RCNN algorithms [1]. Their implementation involved two types of datasets: One dataset, which had pre-labelled images and the other one is a set of images, which were labelled manually. The system also made use of non-maximum suppression to reduce duplicate detections and enhance precision.

Pavinder Yadav et al.(2023) identified gaps between existing technologies for weapon detection. They examined and classified the strengths and shortcomings of several existing algorithms using classical machine learning and deep learning approaches, employed in the detection of different kinds of weapons [3]. This included a comparative analysis of feature extraction methods such as HOG and SIFT, to identify the most effective techniques. A conclusion was drawn with deep learning techniques beating the traditional machine learning techniques in terms of speed and accuracy.

Muhammad Tahir Bhatti et al.(2023) developed a system using open-source deep learning algorithms like VGG16,Inception-V3,InceptionResnetV2, SSDMobileNetV1,FasterRCNNInceptionResnetV2(FRIRv2) and YOLOv4 to identify weapons in an accurate way in video feeds [2].

Muhammad Ekmal Eiman Quyyum and Mohd Haris Lye Abdullah(2023) proposed a system that aims to develop and evaluate the use the deep neural network for weapon detection in surveillance videos. This includes the use of YOLOv3 with Darknet-53 as feature extractor which is used for detecting two types of weapons namely pistol and knife. The YOLOv3 Darknet-53 is further improved by optimizing the network backbone and this is achieved by adding a fourth prediction layer and customizing the anchor boxes in order to detect the smaller objects [4]. Model was evaluated with help of the Sohas weapon detection dataset and showed promising results in precision, recall, and mean average precision(mAP).

Shehzad Khalid et al.(2023) proposed a state-of-the-art system using a deep learning model YOLO V5 for weapon detection, sufficiently resilient in terms of affine, rotation, occlusion, and size. The research showed impressive results in detection accuracy by using instance segmentation or Pixel level segmentation with Mask-RCNN. The system achieved the detection accuracy (DC) of 90.66% and 88.74% Mean intersection over union(mIoU) [5]. Their purposed methodology combined different data augmentation and preprocessing methods to improve the accuracy of the proposed weapon detection system.

The system presented by **M. Milagro Fernandez-Carrobles et al.(2019)** used the Faster R-CNN methodology and compared two CNN architectures of GoogleNet and SqueezeNet. This comparison achieved an 85.44% AP50 for gun detection and 46.68% AP50 for knife detection [7].

3. Methodology:

Proposed Solution:

To address the need for real-time weapon detection, our solution involves convoluted neural networks to accurately identify weapons in surveillance cameras. The system utilizes a YOLOv5-based object detection model which is efficiently trained on a unique dataset of weapon images to ensure better performance.

The solution is divided into multiple steps listed as follows:

- o Data acquisition
- o Model Selection
- o Training of model
- o Streamlit interface implementation and testing

1) Data Acquisition:

Data acquisition is a crucial step in detection systems. It ensures accurate prediction of weapon detection model when the class of the image given as input. The stage is further divided into 2 common stages which are creation of dataset and dataset processing.

Dataset creation:

A dataset is fundamentally important in machine learning as it serves as a raw material from which models learn and therefore directly affecting the prediction of the model. The images for the dataset are downloaded from websites like Adobe stock, Pexels and Unsplash which can download high quality images. There are **4390 total unique images** in the dataset out of which 3512 images are training images and 878 images are testing images which means the weapon dataset is divided into 80% training dataset and 20% testing dataset. There are 4 types of classes which are Knife, Pistol, Scissors and Rifle. There are 640 instances of scissors, 720 instances of pistol,440 instances of rifle and 1,760 instances of knife in the dataset

The Figure 2.1 represents the code output which displays the total summary of the number of instances of the images.

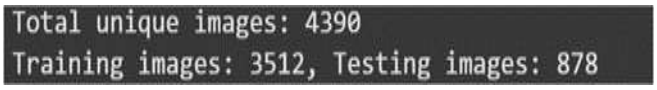

Fig 2.1 Summary of instances

Data Processing:

The downloaded images are initially annotated by drawing a bounding box around the required weapon and labelling them using image annotators like makesense.ai and LabelImg and convert these bounding box dimensions to an XML file and extract them.

Figure 2.2 depicts the first 5 instances of the training and the testing data frames.

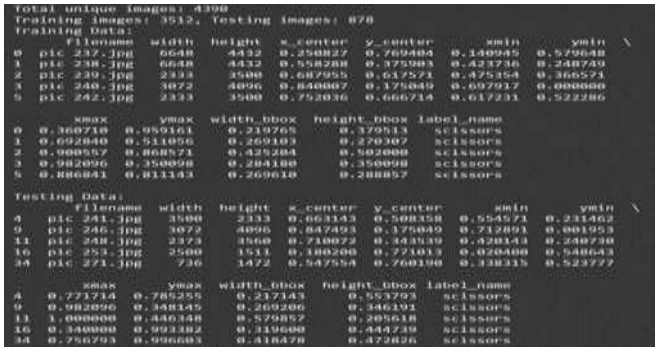

Fig 2.2 Instances of training and testing data frames

1) Model Selection:

We chose Yolov5 as the model which has proven to be much better than other models in terms of FPS and map. YOLO uses a single neural network to predict bounding boxes and class probabilities for objects in an image. YOLO consist of 24 convolutional layers with 2 fully connected layers. Some layers used size 11 of convolutions to decrease the depth of feature map.

2) Training of the model:

The YOLOv5 model has a requirement of a YAML file called data.yaml which consists of label encoding of all the 4 classes and their description in the same order. The yaml file plays a important role in the model training. The figure 2.4 shows the data.yaml file.

Fig 2.4 data.yaml file

The model is fed with the acquired data and is trained with 50 epochs. The main advantage of YOLOv5 model training is that it saves the training progress every second in the form of checkpoints called last.pt and best.pt. The training can be paused at any second and the progress is saved automatically so that when the training is resumed, it starts right from where it had ended before. This is a kind of data backup recovery technique in case of system crashes or exhaustion of processing units where the system shuts down suddenly. The batch size of images is 8 images per batch and the input image resolution is given to be 640X640 pixels. The training iteration goes on till 50 times and saves all the details of that training epoch in a excel sheet.

2) Streamlit interface implementation and testing:

The trained model is saved as a checkpoint file called best.pt where it specifies path of trained model weights and its data and this also contains the weights of the final model that yield the highest accuracy. After training successfully, it is exported to convert the model into a format suitable for it to be deployed outside of the PyTorch framework. The exported model is converted to an **ONNX format.** ONNX (Open Neural Network Exchange) is a popular format used for representing deep learning models, making it portable across different frameworks. The version of ONNX used is Version 12 which ensures compatibility with most modern ONNX- based inference engines. Simplification is done to make the trained weapon detection model faster and more efficient. The ONNX opset value is then set to 12 to ensure that the ONNX operations are compatible with version 12. The trained model is saved as an ONNX file. This .ONNX file is then deployed and made into a user interface using streamlit. The weapon detection model is now embedded in the Streamlit interface.The streamlit interface is created to take in the input from the user. the weapon is then identified and classified and the accuracy percentage along with the type of weapon is displayed. the input can be an image, a video or webcam where each of these inputs is recognized by the system and these options can be chosen accordingly by selecting them from the options made available on the interface. the image/video is selected from the gallery and is uploaded to the system where the system classifies the image/video based on the analysis done on the input. if the input is the camera feed from webcam, then the system keeps on analyzing the feed continuously for the detection of the weapon. if any weapon is detected, an alert message is popped up on the screen that provides the type of weapon and the accuracy score of prediction.

Streamlit integrates numpy and OpenCV for object detection and this process begins with embedding the trained model into the streamlit interface. First, the input image is resized to 40x640 pixels using OpenCV to match the input's size. Then the model makes its predictions and applies bounding boxes and labels for the detected object. Non-Maximum Suppression is used for suppressing false positives by using the confidence score, ensuring that only the most accurate values are retained. The bounding boxes of high confidence are drawn around the detected image as green boxes using OpenCV., showing what class the object belongs to, along with the bounding box and accuracy score. Streamlit makes the whole process interactive, allowing users to upload images or videos and get real-time feedback on the detection results.

Flowchart:

The representation of the system methodology in terms of a flowchart is given below.

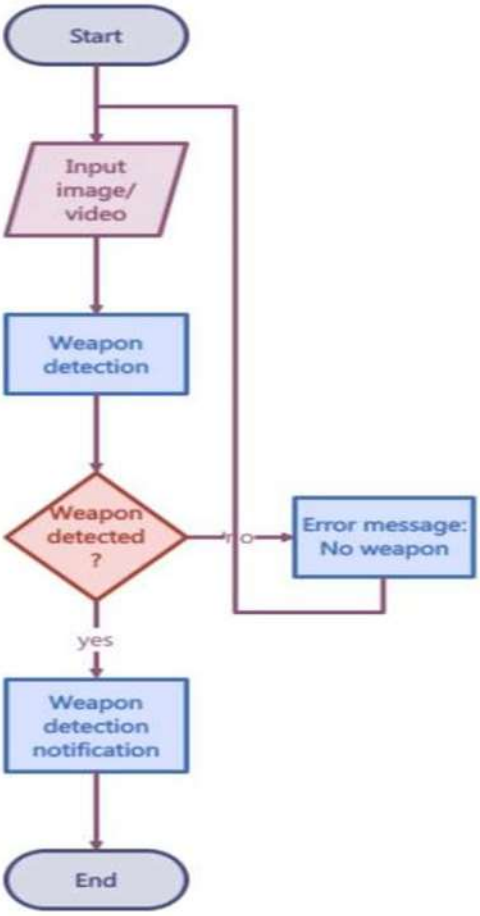

Figure 2.5 System methodology

In Fig 2.5, the model begins by taking the input and then processes the provided input to detect any weapons. If a weapon is found, then a notification/alert is sent to the user. If no weapons were detected, an error message is displayed showing that no weapon was found. The process either loops back for another input or ends after providing the necessary output.

4. Results and discussion:

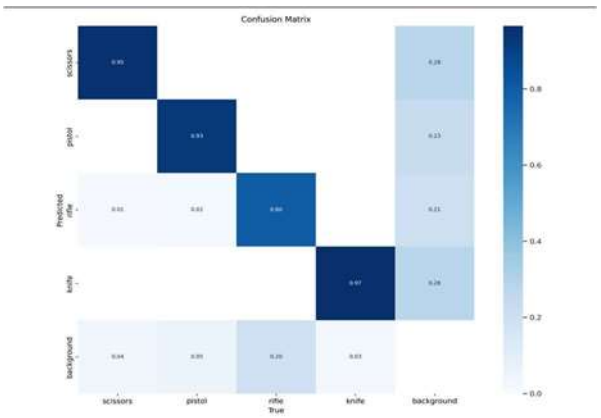

Fig 3.1 Confusion matrix

Fig 3.1 shows a confusion matrix of the model classifying different weapon categories, such as scissors, pistol, rifle, knife. The row represents the predicted classes. The column represents the true (actual) classes. The diagonal values represent the correct classifications. The off-diagonal values represent misclassifications. The matrix describes High Accuracy for Knives (97%) and Scissors (95%), good Accuracy for Pistols (93%) and moderate Accuracy for Rifles (80%). Some weapons (rifle: 21%, pistol: 23%) are misclassified as "background." The model only correctly detects rifles 80% of the time. Also, weapons pistols (0.02) and scissors (0.01) are misclassified. This suggests the model struggles with distinguishing small weapons. And there is very high accuracy for scissors (95%) and knife (97%).

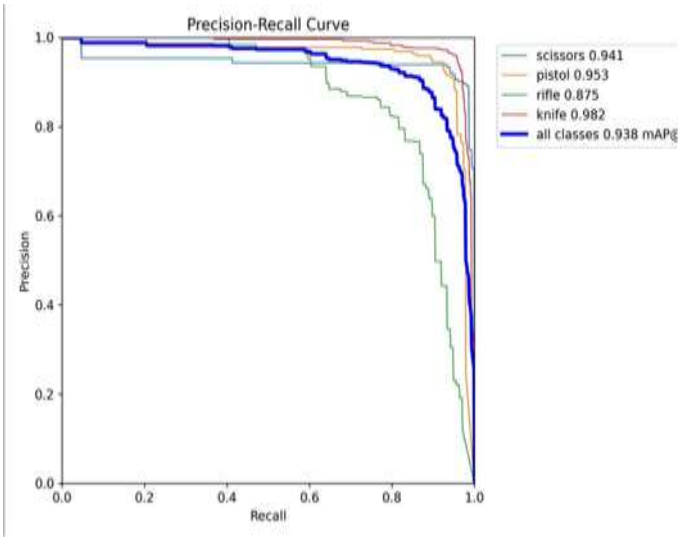

Fig 3.2 Precision-Recall curve

Fig 3.2 represents the PR curve. It plots Precision on the Y- axis against Recall on the X-axis. Scissors (0.941): This indicates that the model has a high precision and recall for detecting scissors. Pistol (0.953): This indicates the system has better performance for pistol detection than scissors. Rifle (0.875): This indicates this class has the lowest score, meaning the model struggles more with detecting rifles compared to other weapons. Knife (0.982): This indicates the best-performing class, with the highest precision-recall score, indicating very accurate knife detection. The PR curves for knife and pistol indicating that the objects are detected with high precision and recall.

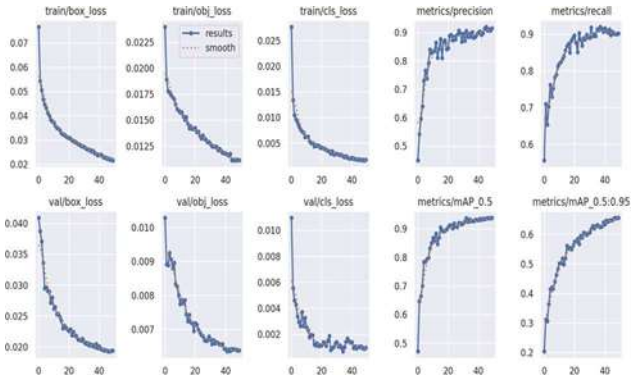

Fig 3.3 Error plot

The error plot in Fig 3.3 contains multiple subplots that represent the training and validation performance of a YOLO object detection model. The train/box_loss measures how well the model predicts bounding box locations. The train/obj_loss measures the confidence of object detection in anchor boxes. The train/cls_loss measures how well the model classifies detected objects into correct categories. The val/box_loss is similar to train/box_loss but on the validation dataset. The val/obj_loss Indicates the ability of the model to detect real objects in validation data. The val/cls_loss indicates model's classification accuracy on validation data. The metrics/precision measures how many detected objects are correct. The metrics/recall measures how many actual objects were detected. The metrics/mAP_0.5 measures how well the model detects objects with IoU > 0.5. The metrics/mAP_0.5:0.95 evaluates model accuracy across multiple IoU thresholds (0.5 to 0.95).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	epoch	train/box_loss	train/obj_loss	train/cls_loss	metrics/precision	metrics/recall	metrics/mAP_0.5	metrics/mAP_0.5:0.95	val/box_loss	val/obj_loss	val/cls_loss	x/lr0	x/lr1	x/lr2
2	0	0.07595	0.02877	0.02765	0.44854	0.55545	0.47548	0.20379	0.04388	0.01026	0.01095	0.07086	0.0032257	0.0032257
3	1	0.05438	0.02622	0.0134	0.54073	0.7089	0.64644	0.31206	0.03707	0.008073	0.005549	0.05937	0.0065272	0.0065272
4	2	0.05057	0.02771	0.01428	0.59575	0.6528	0.66385	0.30634	0.03766	0.008039	0.004835	0.0396733	0.0095966	0.0095966
5	3	0.04668	0.02639	0.009582	0.63979	0.70185	0.70093	0.36301	0.03579	0.006466	0.004281	0.039406	0.009406	0.009406
6	4	0.04426	0.02413	0.008475	0.72965	0.78128	0.78285	0.41297	0.029456	0.006542	0.003854	0.039406	0.009406	0.009406
7	5	0.04182	0.02203	0.008058	0.76521	0.72905	0.78586	0.42074	0.029639	0.007905	0.002908	0.039208	0.009208	0.009208
8	6	0.04122	0.02095	0.007879	0.73784	0.75102	0.79349	0.41627	0.02905	0.006596	0.002636	0.03901	0.00901	0.00901
9	7	0.03903	0.02014	0.007705	0.79103	0.78128	0.79884	0.43887	0.028959	0.006292	0.002585	0.038812	0.008812	0.008812
10	8	0.03837	0.02004	0.007642	0.83959	0.78948	0.83088	0.46278	0.027161	0.006512	0.002626	0.038614	0.008614	0.008614
11	9	0.03768	0.01987	0.006455	0.82791	0.81306	0.84948	0.48282	0.028019	0.006022	0.002553	0.038416	0.008416	0.008416
12	10	0.03723	0.01974	0.006396	0.83086	0.81963	0.86773	0.50111	0.026214	0.007743	0.002503	0.038218	0.008218	0.008218
13	11	0.03568	0.01952	0.006266	0.83458	0.82335	0.85713	0.50131	0.026475	0.007866	0.002018	0.03802	0.00802	0.00802
14	12	0.03504	0.01936	0.00541	0.86427	0.8357	0.88226	0.5183	0.025295	0.007572	0.001768	0.037822	0.007822	0.007822
15	13	0.03462	0.01932	0.005006	0.83012	0.84018	0.84634	0.49824	0.025326	0.007855	0.001936	0.037624	0.007624	0.007624
16	14	0.03396	0.01929	0.005081	0.84334	0.8547	0.86932	0.5195	0.024816	0.007436	0.001593	0.037426	0.007426	0.007426
17	15	0.03285	0.01894	0.004985	0.87612	0.86712	0.90454	0.56133	0.024221	0.007237	0.001207	0.037228	0.007228	0.007228

Fig 3.4 Training and Validation metrics

Fig 3.4 represents the training and validation metrics for a YOLO weapon detection model, across 16 epochs. Each row corresponds to an epoch, and the columns track various losses and performance metrics. The training losses include train/box_loss, which measures how well the model predicts bounding box coordinates, train/obj_loss, which measures object score confidence, and train/cls_loss, which evaluates classification accuracy and similarly. Performance metrics such as precision, recall, and mean average precision (mAP) improve steadily, with precision increasing from 0.44 to 0.87, recall improving from 0.55 to 0.86, mAP_0.5 rising from 0.47 to 0.90, and mAP_0.5:0.95. Validation losses, including val/box_loss, val/obj_loss, and val/cls_loss, also decrease, indicating better generalization, with val/cls_loss dropping significantly from 0.01095 to 0.00107, showing improved classification accuracy. Overall, the consistent decrease in losses and improvements in precision, recall, and mAP suggest that the YOLO model is training effectively and achieving strong object detection performance without significant overfitting.

Streamlit App for Weapon Detection:

Fig 3.5 Streamlit homepage

Fig3.5 is the homepage of the weapon image detection interface built with Streamlit. The user encounters this page before getting to choose the kind of detection (image or video) to be done. Image detection allows users to upload an image for object detection. Webcam detection enables real-time object detection using a webcam. Video detection processes a video file for object detection.

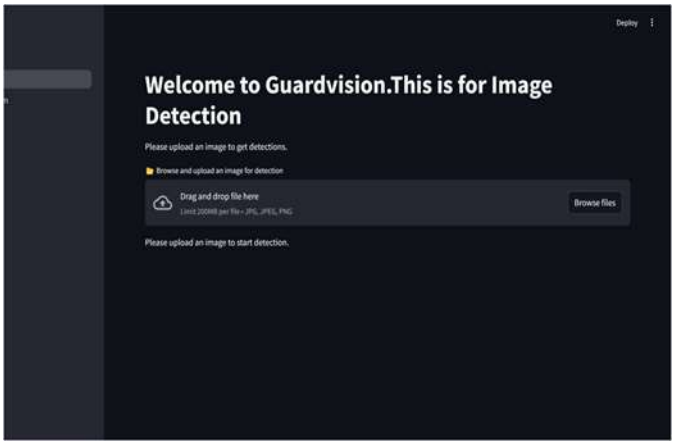

Fig 3.6 Image recognition interface

Fig 3.6 shows that here the user is expected to input the image needed for recognition. This page is designed to allow users to upload an image and perform weapon detection, displaying the detected objects and their confidence scores.

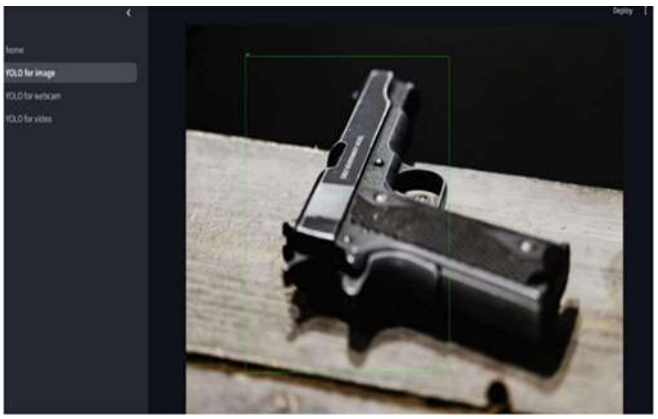

Fig 3.7 Image used for Weapon Detection

In Fig 3.7 the weapon detection system has successfully identified a weapon (pistol) in the uploaded image, marking it with a bounding box. This demonstrates the system's ability to detect objects and highlight them visually. A green bounding box is drawn around the detected object. This indicates that the trained model has identified the object within the image.

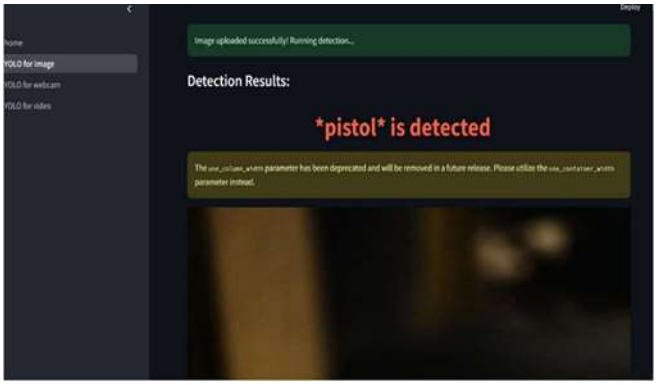

Fig 3.8 Image Detection Success

Fig 3.8 shows the detection results of an uploaded image, successfully identifying a "pistol" as the object detected. The result is displayed prominently to confirm the detection outcome. The text "*pistol* is detected" confirms that the trained model successfully identified a firearm (pistol) in the image.

Fig 3.9 Webcam Weapon Detection

Fig 3.9 shows the detection results of a video successfully identifying a "scissors" as the object detected using a webcam. The result is displayed prominently to confirm the detection outcome. The detected object is labelled as "scissors". The confidence score is 69%, indicating that the model is confident in its classification.

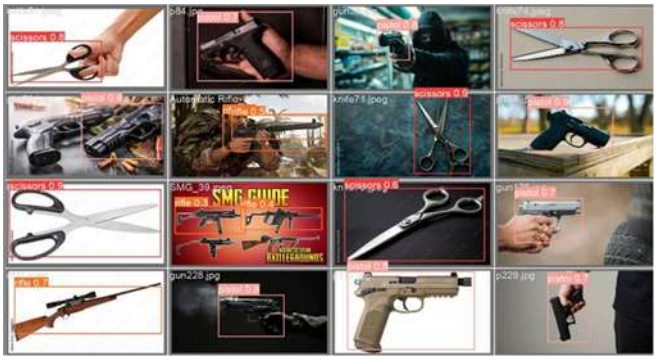

Fig 4.0 Output of an object detection model

Fig 4.0 shows, Scissors: several images contain scissors, detected with confidence scores between 0.6 - 0.9. Pistols

Detected in multiple images, with scores ranging from 0.7 -

0.9. Rifles: Some images are labelled as "rifle" with confidence levels from 0.3 - 0.7. Some rifle detections have lower confidence (0.3-0.5), suggesting possible misclassification.

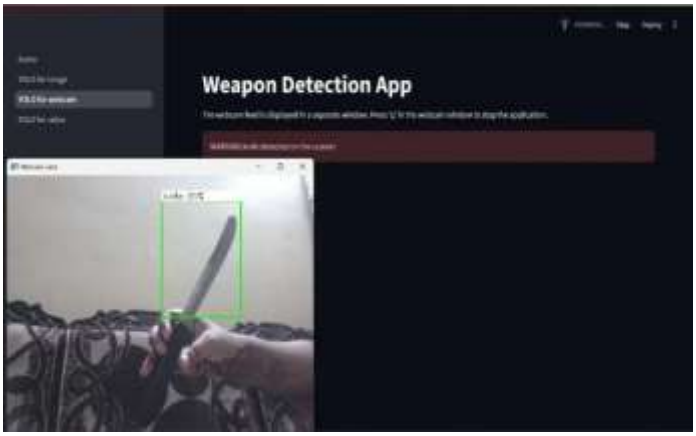

Fig 4.1 Interface of Weapon Detection App

Fig 4.1 shows the interface of a Weapon Detection App that is running to identify weapons in a live webcam feed. Here In the Webcam Feed window, the model has detected a knife with 91% confidence. A green bounding box is drawn around the detected object, labelling it as "knife: 91%". "WARNING: knife detected on the screen!".

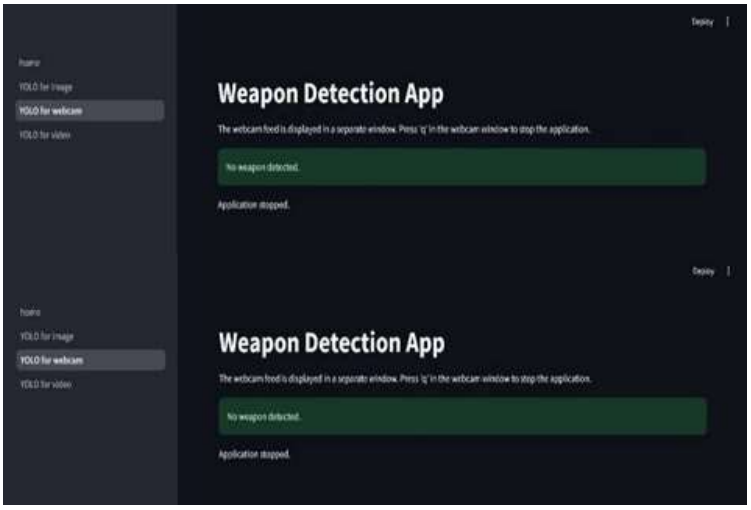

Fig 4.2 Weapon Detection App - No Weapon Detected

Fig 4.2 shows the interface displays a message: "No weapon detected." This message is shown inside a green notification bar, indicating that the model did not identify any objects classified as weapons in the video stream.

5. CONCLUSION AND FUTURE ENHANCEMENTS:

Guard Vision utilizes various features of the YOLOv5 model and OpenCV library in detecting harmful and threatening weapons. The system developed takes in various forms of inputs from the users and provides accurate detection of weapons, helping automate the detection process by enhancing real-time detection accuracy. The detection notification aspect helps keep the user in loop by providing the required information about the weapon as soon as the detection occurs.

While the current system efficiently performs the detection, further enhancements can improve its accuracy and performance. Integrating cloud-based storage for detection provides an opportunity for various sectors to enhance their

References

1. Harsh Jain, A. Vikram, Mohana, A. Kashyap, and A. Jain, "Weapon Detection using Artificial Intelligence and Deep Learning for Security Applications," *Proc. Int. Conf. on Electronics and Sustainable Communication Systems (ICESC 2020)*, Bengaluru, Karnataka, India, 2020,

2. M. T. Bhatti, M. Aslam, M. G. Khan, and M. J. Fiaz, "Weapon Detection in Real-Time CCTV Videos Using Deep Learning," *IEEE Access*, vol. 9, pp. 5678-5688, Feb. 2021.

3. P. Yadav, N. Gupta, and P. K. Sharma, "A Comprehensive Study Towards High-Level Approaches for Weapon Detection Using Classical Machine Learning and Deep Learning Methods," *Int. J. of Scientific Computing*, Aug. 2022.

4. M. E. E. Quyyum and M. H. L. Abdullah, "Weapon Detection in Surveillance Videos Using Deep Neural Networks," in *Proc. MECON 2022*, M. Y. bin Alias, Ed., AER 214, pp. 183–195, 2023.

5. S. Khalid, O. C. Edo, A. Waqar, I. T. Tenebe, and H. U. A. Tahir, "Title of the Paper," *Proc. 2023 Int. Conf. IT Innovation and Knowledge Discovery (ITIKD)*, Islamabad, Pakistan, 2023.

6. T. S. S. Hashmi, N. U. Haq, M. M. Fraz, and M. Shahzad, "Application of Deep Learning for Weapons Detection in Surveillance Videos," *Proc. 2021 Int. Conf. on Digital Technologies (ICoDT)*, Islamabad, Pakistan, May 2021.

7. M. M. Fernandez-Carrobles, O. Deniz, and F. Maroto, "Gun and Knife Detection Based on Faster R-CNN for Video Surveillance," in *Proc. IbPRIA 2019*, A. Morales et al., Eds., LNCS 11868, pp. 441–452, 2019. Springer Nature Switzerland AG, 2019.

8. S. Ahmed, M. T. Bhatti, M. G. Khan, B. Lövström, and M. Shahid, "Development and Optimization of Deep Learning Models for Weapon Detection in Surveillance Videos," *Appl. Sci.*, vol. 12, no. 12, pp. 5772, 2022.

9. D. Qi, W. Tan, Z. Liu, Q. Yao, and J. Liu, "A Dataset and System for Real-Time Gun Detection in Surveillance Video Using Deep Learning," in Proc. IEEE Int. Conf. on Multimedia and Expo (ICME), Shenzhen, China, pp. 1–6, 2021.

10. A. Egiazarov, F. M. Zennaro, and V. Mavroeidis, "Firearm Detection via Convolutional Neural Networks: Comparing a Semantic Segmentation Model Against End-to-End Solutions," in Proc. IEEE Int. Conf. on Image Processing (ICIP), Abu Dhabi, United Arab Emirates, pp. 123–127, 2020.

11. K. Akhila and K. R. Ahmed, "Real-Time Deep Learning Weapon Detection Techniques for Mitigating Lone Wolf Attacks," in Proc. IEEE Int. Conf. on Advanced Video and Signal-Based Surveillance (AVSS), Sydney, Australia, pp. 45–52, 2024.

12. P. Yadav, N. Gupta, and P. K. Sharma, "Robust Weapon Detection in Dark Environments Using YOLOv7-DarkVision," Digital Signal Processing, vol. 145, pp. 104342, April 2024.

13. A. Thakur, A. Shrivastav, R. Sharma, T. Kumar, and K. Puri, "Real-Time Weapon Detection Using YOLOv8 for Enhanced Safety," in Proc. IEEE Int. Conf. on Image Processing (ICIP), Singapore, pp. 1025–1030, 2024.

14. M. M. Fernandez-Carrobles, O. Deniz, and F. Maroto, "Gun and Knife Detection Based on Faster R-CNN for Video Surveillance," in Proc. Iberian Conf. on Pattern Recognition and Image Analysis (IbPRIA), Madrid, Spain, pp. 441–452, 2019.

15. S. B. Mane, "Weapon Detection and Classification Using Deep Learning," J. Eng. Technol. Ind. Appl., vol. 10, no. 47, pp. 19–26, May/June 2024.

16. R. Sharma, T. Kumar, and K. Puri, "Real-Time Weapon Detection Using YOLOv8 for Enhanced Safety," in Proc. IEEE Int. Conf. on Image Processing (ICIP), Singapore, pp. 1025– 1030, 2024.