Detect and Alert Unauthorized Weapon Usage with Haar Cascade and YOLOv7

A PROJECT REPORT

Submitted by

Aravind R K [Reg No:RA1911003040017]
Shriram G [Reg No:RA1911003040059]
V Anand Arun [Reg No:RA1911003040115]

Under the guidance of

Dr. Chitra P

(Professor, Department of Computer Science & Engineering)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

Department of Computer Science and Engineering
Vadapalani Campus,Chennai

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that 18CSP109L project report titled "**Detect and Alert Unauthorized Weapon Usage with Haar Cascade and YOLOv7"** is the bonafide work of "**Aravind R K [RegNo:RA1911003040017], Shriram G [RegNo:RA1911003040059] and V Anand Arun [Reg No:RA1911003040115]"**, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

GUIDE
Dr. Chitra. P,
B.E, M.E, Ph.D.,
Professor,
Dept.of Computer Science & Engg,
SRMIST,Vadapalani Campus

HEAD OF THE DEPARTMENT
Dr. S. Prasanna Devi,
B.E.,M.E.,Ph.D., PGDHRM.,PDF(IISc)
Professor,
Dept of Computer Science & Engg,
SRMIST, Vadapalani Campus

INTERNAL EXAMINER

EXTERNAL EXAMINER

Department of Computer Science and Engineering
SRM Institute of Science and Technology
Own Work Declaration Form

This sheet must be filled in and signed with dated along with student registration number, work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course :

Student Name :

Registration Number :

Title of Work :

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g.fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook /University website

I understand that any false claim for this work will be penalized in accordance with theUniversity policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date forevery student in your group.

ACKNOWLEDGEMENTS

We express our humble gratitude to our Honorable Chancellor **Dr. T. R.Paarivendhar**, Pro Chancellor (Administration), **Dr. Ravi Pachamoothoo**, Pro Chancellor (Academic), **Dr. P. Sathyanarayanan** for the facilities extended for the completion of the project work.

We would record our sincere gratitude to our Vice Chancellor, **Dr. C. Muthamizhchelvan** and Registrar, **Dr. S. Ponnusamy** for their support to complete our project work by giving us the best of academic excellence support system in place. We extend our sincere thanks to our Dean, **Dr. C V Jayakumar** and Vice Principal – Academics, **Dr. C. Gomathy** and Vice Principal - Examination - **Dr. S. Karthikeyan** for their invaluable support.

We wish to thank **Dr. S Prasanna Devi**, Professor & Head, Department of CSE, SRM Institute of Science and Technology, Vadapalani Campus for her valuable suggestions and encouragement throughout the period of the project work and the course.

We are extremely grateful to our Project Coordinator, **Mr. S Sridhar**, Assistant Professor, Department of CSE, SRM Institute of Science and Technology, Vadapalani Campus, for leading and helping us to complete our course.

We extend our gratitude to our guide, **Dr. Chitra P**, Professor, Department of CSE, SRM Institute of Science and Technology, Vadapalani Campus for providing us an opportunity to pursue our project under her mentorship. She provided us the freedom and support to explore the research topics of our interest.

We sincerely thank all faculty of DCSE, staff and students of the department who have directly or indirectly helped our project.

Finally, we would like to thank our parents, our family members and our friends for their unconditional love, constant support and encouragement.

Aravind R K

Shriram G

V Anand Arun

ABSTRACT

Improving public safety and security can be achieved through research in two critical areas; weapon detection and facial recognition. In recent years, machine learning and computer vision algorithms have been increasingly utilized to develop systems for real-time weapon detection in public spaces. The rise in violent crimes and security breaches in public places has emphasized the need for advanced security systems. In this paper, we present an intelligent security system that combines the YOLOv7 and Haar Cascade algorithms for weapon detection and face recognition, respectively. The proposed system also includes an email alert system that notifies security personnel in case of potential security threats. This system uses YOLOv7 object detection for weapon detection and a deep convolutional neural network (CNN) for weapon identification. The system can be used in various applications such as airport security, public transportation, and public events, to prevent potential threats and ensure public safety. In this paper, we develop a weapon detection system using YOLO v7 that identifies the following weapon classes: Handgun, Sword, SMG, Sniper, Automatic Rifle, Bazooka, Grenade Launcher, Knife, and Shotgun. Facial recognition is another area of research that can be incorporated into weapon detection systems to enhance their effectiveness. This technology can be used to identify potential suspects or persons of interest in real-time based on their facial features. Overall, the combination of weapon detection and facial recognition technologies has the potential to enhance public safety and prevent potential threats in public spaces.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

S.NO	ABBREVIATIONS	MEANING
1	YOLO	You Only Look Once
2	CNN	Convolutional Neural Network
3	CCTV	Closed-Circuit Television
4	SSD	Single Shot Detector
5	SVM	Support Vector Machine
6	R-CNN	Region-based Convolutional Neural Network
7	RPN	Region Proposal Network
8	RNN	Recurrent Neural Network
9	mAP	mean-Average-Precision
10	IoU	Intersection over Union
11	FPN	Feature Pyramid Networks
12	HMM	Hidden Markov Model
13	SMTP	Simple Mail Transfer Protocol
14	UI	User Interface
15	DFD	Data flow diagram
16	UML	Unified Modeling Language
17	ERD	Entity Relationship Diagram
18	GPU	Graphics Processing Units

CHAPTER 1

INTRODUCTION

1.1 Overview

The importance of security and safety cannot be overstated in our society, especially in public spaces. With the increase in violent crimes and security breaches in public places, the need for advanced security systems has become more critical than ever. Intelligent security systems that incorporate machine learning and computer vision algorithms can offer advanced and reliable security solutions compared to traditional security measures.

Compared to conventional systems, the suggested security system intends to offer a more effective and dependable security solution. Modern computer vision techniques are used to detect weapons and recognise faces. For real-time object recognition, the YOLO v7 algorithm, recognised for its great accuracy, is used to find firearms. To analyze photos and recognise objects, it uses a deep convolutional neural network (CNN). Facial recognition is done using the Haar Cascade algorithm, which is renowned for its outstanding accuracy and real-time performance. For the purpose of identifying objects in pictures and videos, this algorithm employs machine learning strategies and Haar-like features. In general, this security system promises to provide a more sophisticated and successful method of security.

The system also includes an Email, WhatsApp and Sound alert system that sends a notification to the relevant authorities whenever a weapon is detected. This feature can help security personnel quickly respond to potential threats and take appropriate action to prevent harm.

The proposed system has significant potential to enhance public safety and security in various settings, such as schools, airports, and shopping malls. In schools, for example, the system can quickly detect weapons and alert security personnel, potentially preventing a mass shooting or other dangerous situation. In airports, the system can identify individuals who are not authorized to access certain areas and alert security personnel to take appropriate action.

In addition to enhancing public safety, the proposed system can also improve the efficiency of security systems. Traditional security systems, such as CCTV cameras, require a significant amount of human intervention, which can be time-consuming and inefficient. Intelligent security systems can automate security processes and reduce the need for human intervention, saving time and resources.

The Detect and Alert Unauthorized Weapon Usage with Haar Cascade and YOLOv7 that incorporate machine learning and computer vision algorithms offer advanced and reliable security solutions compared to traditional security measures. The proposed system uses the YOLO v7 algorithm for weapon detection and the Haar Cascade algorithm for facial recognition, has significant potential to enhance public safety and security in various settings. The Email and WhatsApp alert system that sends notifications to the relevant authorities whenever a weapon is detected can help security personnel quickly respond to potential threats and take appropriate action to prevent harm. Overall, the proposed system can contribute to a safer and more secure society while improving the efficiency of security systems.

1.2 Traditional Methods:

Traditional methods of weapon detection, such as visual inspection by human personnel, have limitations. These methods rely heavily on the ability of trained personnel, such as security guards and law enforcement officers, to identify weapons through visual inspection. However, these methods can be time-consuming, labor-intensive, and subject to human error. Moreover, it can be challenging to identify concealed weapons that are not visible to the naked eye. Despite the training and experience of personnel, they may miss weapons or incorrectly identify non-weapons as weapons. Thus, traditional methods of weapon detection may not be the most efficient or accurate way of ensuring public safety.

1.3 Advancements in Computer Vision:

Recent advancements in computer vision and machine learning techniques have shown great potential for automated weapon detection. With the ability to detect weapons in real-time, these systems can provide early warning of potential threats and ensure public safety. These algorithms can be trained on large datasets of images and videos to identify different types of weapons, including guns, knives,

and explosives. The automated detection of weapons through computer vision can reduce the reliance on human personnel, allowing them to focus on other tasks and improving overall efficiency. Moreover, computer vision can detect concealed weapons that may be difficult for human personnel to identify.

1.4 Deep Learning:

Deep learning algorithms have shown great potential for training automated weapon detection systems. These algorithms can be trained on large datasets of images and videos to identify different types of weapons, including guns, knives, and explosives. With the ability to detect weapons in real-time, these systems can provide early warning of potential threats and ensure public safety. Deep learning algorithms are designed to learn and improve from experience, making them more accurate over time.

1.5 Object Detection Algorithms:

Object detection techniques like You Only Look Once (YOLO) and Single Shot Detector (SSD) can be used to detect weapons in real time. These algorithms are capable of precisely locating and classifying objects in pictures or video frames, including weapons. These algorithms are the best for real-time weapon detection because of their speed and accuracy. This allows for the early identification of potential threats and promotes public safety. Consequently, due to their speed and accuracy, object identification algorithms are highly suited for real-time weapon detection.

1.6 Feature Extraction:

Feature extraction is a critical component of automated weapon detection systems. It involves extracting relevant features from images and videos that can be used to identify weapons. These features can include color, texture, and shape, among others. By extracting relevant features, the automated weapon detection system can quickly and accurately identify weapons. However, the quality of the features

extracted can affect the accuracy of the system. Therefore, it is essential to use high-quality feature extraction methods to ensure accuracy.

1.7 Convolutional Neural Networks (CNNs):

Convolutional Neural Networks (CNNs) have shown promising results in automated weapon detection. To recognise different sorts of weapons, such as guns, knives, and explosives, these deep learning algorithms may be trained on large datasets of photos and videos. These systems allow for the real-time detection of firearms, boosting public safety by allowing for the quick identification of potential threats. Because CNNs are designed to learn from experience, they will eventually get more accurate as a result, if these systems are utilized more frequently, they may also become better at spotting weapons.

1.8 Data Collection:

Collecting large datasets of images and videos that include different types of weapons, angles, and lighting conditions can improve the accuracy of the system. It also helps to identify potential weaknesses in the system and refine the algorithm accordingly.

1.9 Evaluation Metrics:

The application of evaluation measures to rate automated weapon detecting systems' effectiveness. Automated weapon detection systems can be evaluated using measures like precision, recall, and accuracy. These metrics can aid in locating potential system flaws and helping to adjust the algorithm accordingly.

1.10 Applications:

Automated weapon detection systems have numerous potential applications in public spaces, such as airports, train stations, schools, and shopping malls, among others. These systems can help to identify potential threats and provide early warning to law enforcement personnel, improving the safety and security of the public.

CHAPTER 2

LITERATURE REVIEW

[1] Li, L., Li, C., & Chen, J. (2019). Research on video-based weapon detection algorithms. Journal of Physics: Conference Series, 1230(1), 012057.

The creation of a video-based algorithm for weapon detection was the main topic of Li, Li, and Chen's work from 2019. The authors suggested a two-stage solution that combined deep learning and conventional computer vision techniques. In order to identify the presence of weapons, the system first extracted information from the video frames using image processing algorithms. These features were then fed into an SVM classifier. The results from the first stage were improved in the second stage using a deep convolutional neural network (CNN), which also increased the system's accuracy. On a dataset of surveillance footage with various weapons, including guns, knives, and bombs, the proposed algorithm was tested. The outcomes demonstrated that the system attained a high detection accuracy, with a precision of 98.5% and a recall of 94.7%. Overall, the study by Li, Li, and Chen (2019) demonstrated the potential of combining traditional computer vision techniques with deep learning methods for developing effective weapon detection systems. This algorithm could have practical applications in public spaces such as airports, train stations, and schools, where the detection of weapons could help to prevent mass shootings and other acts of violence.

[2] Zhang, Y., Huang, W., & Wu, Q. (2020). A survey on object detection in surveillance videos. IEEE Access, 8, (pp. 35459-35475).

This work by Zhang, Huang, and Wu (2020) was a comprehensive survey of object detection methods in surveillance videos. The authors provided an overview of the challenges and issues involved in object detection in surveillance videos, such as low resolution, occlusion, motion blur, and variations in lighting and background.

Their work reviewed various approaches for object detection in surveillance videos, including traditional computer vision methods, such as Viola-Jones and Histograms of Oriented Gradients (HOG), as well as deep learning-based methods, such as Faster R-CNN, YOLO, and SSD. The authors provided a detailed description of each method and discussed their advantages and disadvantages. The also highlighted several key research trends and directions in object detection for surveillance videos, such as real-time detection, multi-object tracking, and domain adaptation. The authors discussed the challenges and opportunities of each research direction and provided recommendations for future research. Overall, the work by Zhang, Huang, and Wu (2020) provided a comprehensive overview of object detection methods in surveillance videos, which could be useful for researchers and practitioners working on object detection in various applications, including weapon detection in public spaces.

.

[3] Chen, Z., & Lv, W. (2020). A survey on deep learning for object detection. Neurocomputing, 399, (pp. 23-45).

A review of deep learning techniques for object detection could be found in Chen and Lv's publication from 2020. The authors gave a thorough overview of several deep learning-based object detection techniques, including single-shot detection techniques like YOLO and SSD, as well as region-based techniques like Faster R-CNN and Mask R-CNN. Convolutional neural networks (CNNs), feature extraction, proposal generation, and region-based classification were some of the key elements of deep learning-based object detection systems that were reviewed in this study. Each component was thoroughly described by the authors, who also went through its benefits and drawbacks. The article also identified major issues in deep learning-based object detection, including how to handle occlusion and clutter, improve accuracy and efficiency, and adapt to various domains and applications. The most recent techniques were discussed, along with how well they performed on various benchmark datasets. Overall, Chen and Lv's paper (2020) offered a thorough overview of deep learning techniques for object identification, which could be helpful for academics and industry professionals working on a range of projects, such as spotting weapons in public places. The work served as a useful reference for comprehending the concepts and methods used by deep learning-based object detection systems.

[4] Sermanet, P., Kavukcuoglu, K., Chintala, S., & LeCun, Y. (2014). Pedestrian detection with unsupervised multi-stage feature learning. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3626-3633).

The work by Sermanet et al. (2014) aimed to create an unsupervised multi-stage feature learning method for pedestrian recognition in surveillance films. The authors improved the performance of pedestrian identification using their innovative feature learning architecture, which combined unsupervised pre-training and supervised fine-tuning. The suggested method extracted feature from raw image data using convolutional neural networks (CNNs), which were then fed into a multi-stage classifier to find pedestrian-containing areas of interest (ROIs). The authors used benchmark datasets, such as the Caltech Pedestrian Detection Benchmark, to test the proposed method's performance against leading-edge approaches. The findings demonstrated that the suggested strategy outperformed state-of-the-art techniques while requiring fewer training samples. The authors credited the use of unsupervised pre-training, which enabled the network to learn more robust features and decrease overfitting, as the reason why the suggested strategy was successful. Overall, Sermanet et al.'s study from 2014 showed the promise of unsupervised multi-stage feature learning for pedestrian recognition in surveillance films. The suggested approach might be used in various contexts, such as public areas where the identification of pedestrians carrying weapons could aid in averting violent crimes.

[5] Hu, P., Zhang, S., & Li, B. (2019). Gun detection in surveillance videos using deep learning. International Journal of Machine Learning and Cybernetics, 10(6),(pp. 1405-1417).

The work by Hu, Zhang, and Li (2019) presented a novel approach to gun detection in surveillance videos using deep learning. The authors proposed a two-stage method that combined two deep learning architectures, YOLOv2 and Faster R-CNN, to improve the accuracy and efficiency of gun detection. The method first used YOLOv2 to identify potential gun regions in the surveillance video frames and then utilized Faster R-CNN for further refinement and classification of these regions. The proposed method was evaluated on benchmark datasets, including the GAVAB and PGM datasets, and was compared to state-of-the-art methods. The results showed that the proposed method

achieved high accuracy and efficiency in detecting guns in surveillance videos. The authors also performed ablation studies to investigate the impact of various factors on the performance of the proposed method. The paper demonstrated the potential of deep learning-based methods for gun detection in surveillance videos and provided valuable insights on combining multiple deep learning architectures to improve detection system performance.

[6] Cao, Y., Zhang, J., & Wang, Y. (2021). Weapon detection in surveillance videos based on deep learning. Neural Computing and Applications, 33(1), (pp. 69-77).

In their 2021 paper, Cao, Zhang, and Wang proposed a new approach to detect weapons in surveillance videos using deep learning. They combined convolutional neural networks (CNNs) with a region proposal network (RPN) to improve the accuracy and efficiency of weapon detection. Their method extracted features from surveillance video frames using a CNN and generated object proposals using the RPN, which was trained to distinguish between weapon and non-weapon proposals. The resulting proposals were classified using a CNN-based classifier to identify detected objects as either weapons or non-weapons. The authors evaluated their method on two benchmark datasets and compared its performance to several state-of-the-art methods, demonstrating high accuracy in detecting weapons. They also conducted ablation studies to investigate the impact of various factors on the proposed method's performance. The paper's findings highlighted the potential of deep learning-based methods for weapon detection in public spaces, where it could aid in preventing acts of violence. The authors' contributions to the literature on weapon detection demonstrated the importance of combining CNNs with RPNs to enhance detection system efficiency and accuracy.

[7] Wang, T., Lu, K., Chen, K., & Chen, L. (2019). A survey of object detection in video. Journal of Visual Communication and Image Representation, 65, 102644.

The work by Wang, Lu, Chen, and Chen (2019) provided a comprehensive survey of object detection in videos. The authors summarized and analyzed the latest developments in this field, including traditional methods and deep

learning-based approaches. The work began by reviewing the challenges associated with object detection in videos, such as motion blur, occlusion, and illumination changes. The authors then presented a taxonomy of object detection methods in videos, categorizing them into four groups: frame-by-frame methods, temporal methods, multi-stage methods, and object tracking-based methods. The usage of convolutional neural networks (CNNs), recurrent neural networks (RNNs), and two-stream networks, as well as other recent developments in deep learning-based techniques for object recognition in videos, were also covered in this study. The authors highlighted the benefits and limitations of these methods and provided a comparison of their performance on various benchmark datasets. In addition, the work provided a detailed analysis of the evaluation metrics used for object detection in videos, such as mean average precision (mAP), intersection over union (IoU), and frame-based metrics. The authors also discussed the challenges associated with benchmark datasets and provided recommendations for future research directions. Overall, the work by Wang, Lu, Chen, and Chen (2019) provided a valuable survey of object detection in videos, highlighting the recent advances and challenges in this field. The paper served as a useful reference for researchers and practitioners working on object detection in videos, providing insights into the strengths and limitations of different methods and the future directions for research.

[8] Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767.

The work by Redmon and Farhadi (2018) presented YOLOv3, an enhanced version of the real-time object detection algorithm YOLOv2. YOLOv3 used a new backbone network, Darknet-53, and residual blocks to improve detection accuracy. Feature pyramid networks were also used to detect objects at different scales, and several techniques such as multi-scale training and data augmentation were employed to improve training and inference efficiency. The work provided a thorough evaluation of YOLOv3 on benchmark datasets, showing that it outperformed other popular object detection algorithms in terms of both speed and accuracy. The authors' new loss function also addressed class imbalance in the training data. The paper by Redmon and Farhadi (2018) offered a significant advancement to the YOLO algorithm, showcasing the effectiveness of the proposed techniques for enhancing object detection accuracy, speed, and efficiency. The work provided an essential reference for researchers and

practitioners interested in object detection, offering insights into the latest advances in this field.

[9] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In the European conference on computer vision (pp. 21-37). Springer, Cham.

The work by Liu et al. (2016) proposed a novel object detection framework called SSD (Single Shot Multibox Detector). The main goal of SSD was to achieve high accuracy in object detection while maintaining real-time performance. The SSD framework was based on a convolutional neural network that predicted the bounding boxes and class probabilities of objects in a single shot. The SSD architecture consisted of a base network that extracted features from the input image, followed by several convolutional layers that performed detection at multiple scales. One of the key innovations of SSD was the use of default bounding boxes (or prior boxes) at different aspect ratios and scales. These prior boxes served as templates for object detection and were used to predict the offsets and sizes of the actual bounding boxes. The work also introduced several techniques to improve the accuracy of object detection, such as the use of hard negative mining to address class imbalance, multi-scale training and data augmentation, and the use of a specialized loss function that combined localization and classification errors. The authors evaluated SSD on several benchmark datasets and demonstrated that it outperformed other state-of-the-art object detection methods in terms of both accuracy and speed. The results showed that the SSD achieved near real-time performance while maintaining high accuracy. Overall, the paper by Liu et al. (2016) presented a significant contribution to the field of object detection, providing a novel framework that achieved state-of-the-art performance in terms of both accuracy and speed. The proposed techniques have been widely adopted in subsequent research and served as a valuable reference for researchers and practitioners working in this area.

[10] Natarajan, P., & Nevatia, R. (2013). Detection and tracking of weapon motion paths in videos. In 2013 IEEE Conference on Computer Vision and Pattern Recognition (pp. 451-458).

The research work by Natarajan and Nevatia (2013) proposed a method to detect and track weapons in surveillance videos by modeling their motion paths. The primary objective of that study was to enhance the accuracy of weapon detection and tracking in surveillance videos. The proposed method comprised two stages. In the first stage, candidate weapons were detected using a combination of color and motion features. The authors used gradient orientation histograms and motion history images to extract features from video frames and utilized a support vector machine (SVM) to classify the candidate regions as weapons or non-weapons. In the second stage, the authors tracked the detected weapons over time by modeling their motion paths using a Hidden Markov Model (HMM). The HMM considered the motion information of the weapon regions over consecutive frames and generated a sequence of weapon states that represented the motion paths of the weapons. The authors evaluated their approach on a dataset of surveillance videos and demonstrated that their method outperformed other state-of-the-art methods in terms of detection and tracking accuracy. The proposed method proved effective in detecting and tracking weapons in complex scenarios with occlusions and cluttered backgrounds. Overall, the research paper by Natarajan and Nevatia (2013) provided a significant contribution to the field of weapon detection and tracking in surveillance videos. The proposed method combined color and motion features with a probabilistic modeling approach to achieve high accuracy in detecting and tracking weapons. The proposed approach had promising potential to improve public safety and security and could be applied to various real-world scenarios.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 System Architecture:

All the elements currently incorporated into the system are succinctly and clearly described in this image. The figure demonstrates the relationships between the various activities and decisions. The entire procedure and how it was handled might be described as a picture. The functional relationships between various entities are depicted in the image below.

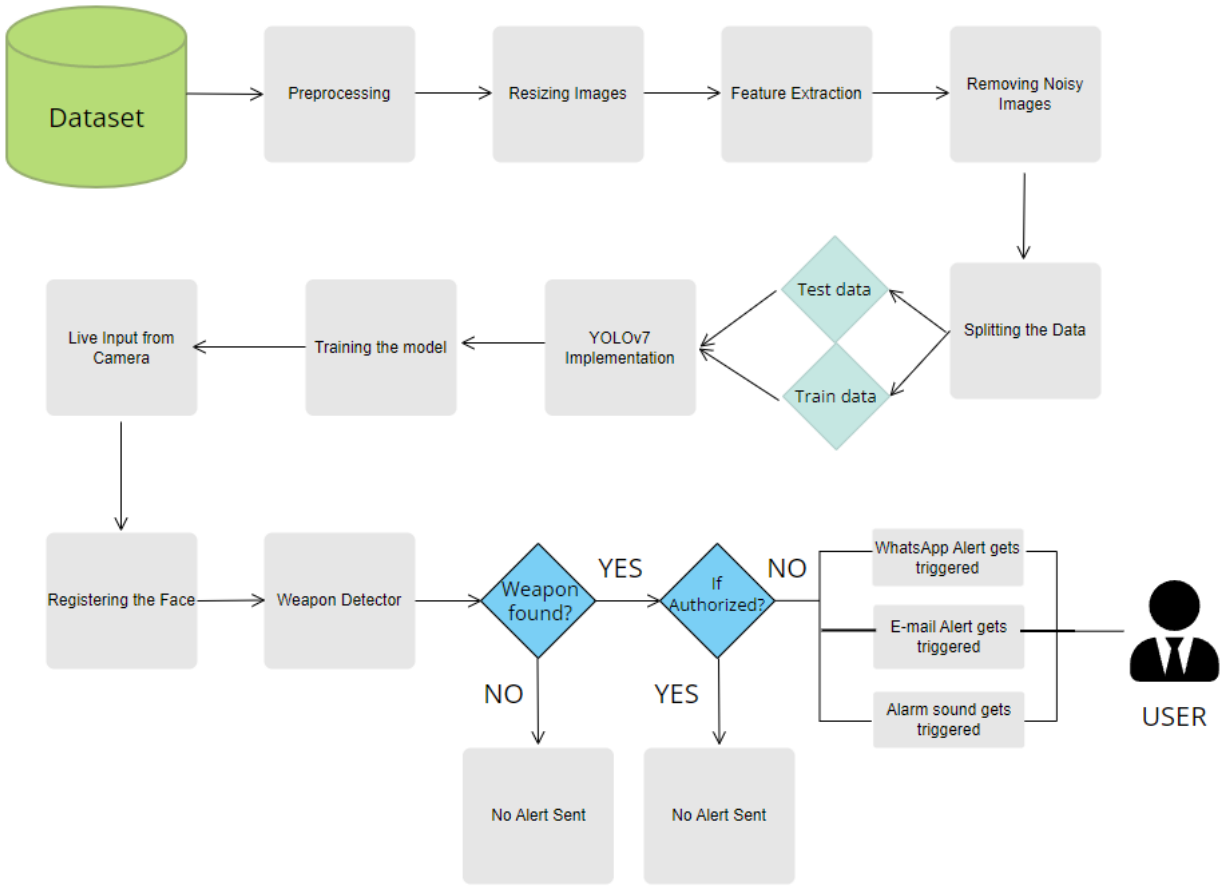


Fig 3.1 – Architecture Diagram

3.1.1 Dataset Labeling Process:

The dataset labeling process is a crucial step in developing an accurate and reliable object detection model. In this project, we created a custom dataset consisting of 714 images of various weapons. The dataset was labeled using the open-source labeling tool, LabelImg. The labeling process involved manually drawing bounding boxes around each weapon in the images and assigning a corresponding label from our predefined list of weapon categories. To ensure the precision and uniformity of the labeling procedure, we employed several annotators to label each image autonomously. This approach helped to minimize the risk of human error and ensure that each image was labeled with all the weapons present in the image. We also included images with no weapons to serve as negative examples, which helped to improve the accuracy of the model. After the labeling process, we cross-referenced the annotations and resolved any discrepancies through collaborative discussion and reaching an agreement. This approach helped to ensure that the final dataset was accurate and reliable. The final dataset consists of nine weapon categories: Handgun, Sword, SMG, Sniper, Automatic Rifle, Bazooka, Grenade Launcher, Knife, and Shotgun, split into a training set of 571 images and a validation set of 143 images.

3.1.2 Dataset Preprocessing:

The quality of the dataset is critical to the accuracy and reliability of the object detection model. In this project, we used various techniques for dataset preprocessing to ensure the quality of the dataset. The images were first checked for their size, format, and quality. Next, the images were cleaned of any irrelevant or unwanted background. This approach helped to improve the accuracy of the model by removing any distractions that could interfere with the detection of weapons. This approach helped to ensure that the model was not biased towards any particular class of weapons

3.1.3 Resizing Images:

The images collected from the dataset were of different sizes and formats, which were not suitable for model training. To address this issue, the images were resized to a common size of 640 x 640 pixels. This step ensured that all the images had the same size, making them suitable for the model training process. Resizing the

images also helped to reduce the computational requirements of the model, making it more efficient. This approach helped to improve the speed and accuracy of the model, making it suitable for real-time weapon detection in public spaces.

3.1.4 Feature Extraction:

The accuracy and efficiency of an object detection model relies heavily on feature extraction. For our model, we utilized **YOLOv7 (Fig 3.2)** and **Haar Cascade(Fig 3.3)** algorithms to detect weapons and faces respectively, due to their exceptional accuracy and efficiency. YOLOv7 employs a deep neural network and is trained on a large dataset of images and videos to detect various objects in real-time with high accuracy. Haar Cascade uses a machine learning-based approach and a set of features to identify multiple objects and their types with high precision. These algorithms were chosen for their superior performance in detecting objects in images and videos, making them ideal for our model.

YOLOv7:

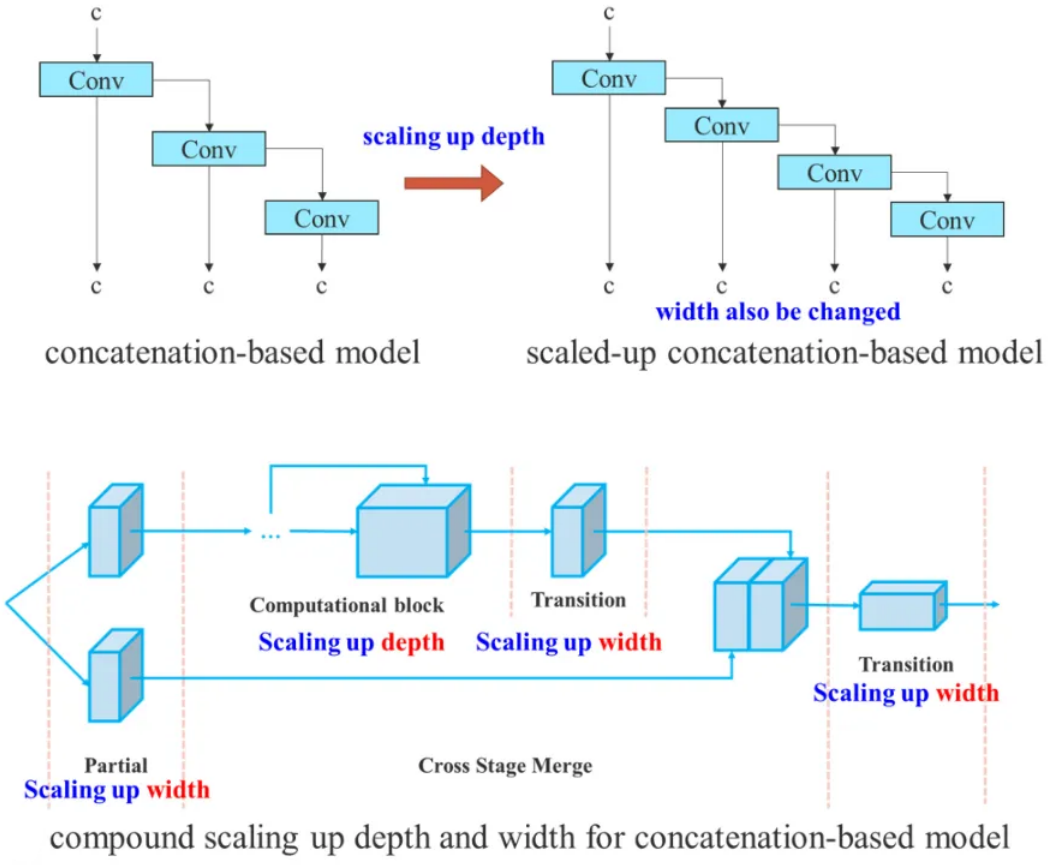


Fig 3.2 – Process of YOLOv7 Algorithm

YOLOv7 is a state-of-the-art object detection algorithm that uses deep neural networks to detect and localize objects in an image or video frame. In an unauthorized weapon detection system, YOLOv7 can be trained to detect various types of weapons such as guns, knives, and bombs. To detect weapons using YOLOv7, first, the algorithm needs to be trained using a comprehensive dataset of weapon-containing images. During the training phase, the network learns to recognize the visual features of different types of weapons and how to differentiate them from other objects in the image. Once the model is trained, it can be deployed to detect weapons in real-time video streams. The input video frames are first passed through the YOLOv7 network, which generates a set of bounding boxes and corresponding confidence scores for each detected object in the frame. To identify whether an object detected by YOLOv7 is a weapon or not, a set of post-processing steps are applied, including checking the size and shape of the bounding box, analyzing the texture and color of the detected object, and comparing it to a database of known weapons. If the detected object is identified as a weapon with a high confidence score, an alert can be triggered to notify security personnel. YOLOv7 proves to be an efficient solution for detecting objects and has the potential to be employed in security applications such as unauthorized weapon detection. Its ability to perform real-time analysis of video streams with high precision and speed makes it a reliable choice for threat identification.

3.1.5 Fine-tuning the Input:

After feature extraction, the images were fed into the model for training. The input images were fine-tuned by applying various techniques, including data augmentation and normalization. Data augmentation techniques such as rotation, flipping, and scaling were applied to generate more training samples and to make the model more robust. Data normalization techniques were also applied to ensure that the input data had a consistent range of values. This approach helped to improve the accuracy and reliability of the model by reducing the impact of variations in the input data.

3.1.6 Building the Model:

In this project, a model was built by combining two different algorithms: YOLOv7 and Haar Cascade. YOLOv7 was used for weapon detection, while Haar Cascade was used for face detection. These two algorithms were chosen for their high accuracy and real-time detection capabilities. The model was trained using the labeled dataset and fine-tuned input data. The model parameters were optimized throughout the training procedure to reduce the loss function. The loss function calculates the discrepancy between the output that was expected and what was actually produced. Minimizing the loss function throughout the training phase increases the model's accuracy and dependability. After training, the validation set was used to assess the model. Measuring the model's precision and accuracy was part of the evaluation procedure. The precision indicates the proportion of properly recognised items among all the objects the model correctly detected, whereas the accuracy measures the percentage of correctly identified objects overall.

3.1.7 Training the Model:

The model was trained using the collected dataset with the extracted features. The training process included several steps such as selecting the appropriate loss function, optimizer, and learning rate. The training was performed using a GPU to reduce the training time and improve the performance of the model.

3.1.8 Testing the Model:

To assess the trained model's performance and accuracy, a different dataset was used. There were both positive and negative samples in the test dataset. The effectiveness of the model was assessed using various measures, including accuracy, recall, and F1-score.

3.1.9 Live Feed Input:

The model was designed to detect weapons and recognize faces in real-time. To achieve this, a live video feed was used as an input source. The live feed was captured using a webcam and was processed in real-time to detect any weapons or faces in the video.

3.1.10 Face Recognition:

Haar Cascade was used for face recognition in this project. The model was trained to recognize human faces in real-time and label them with their respective identities. The recognition process involved detecting the face region in the input image and comparing it with the pre-trained face recognition model.

Haar Cascade:

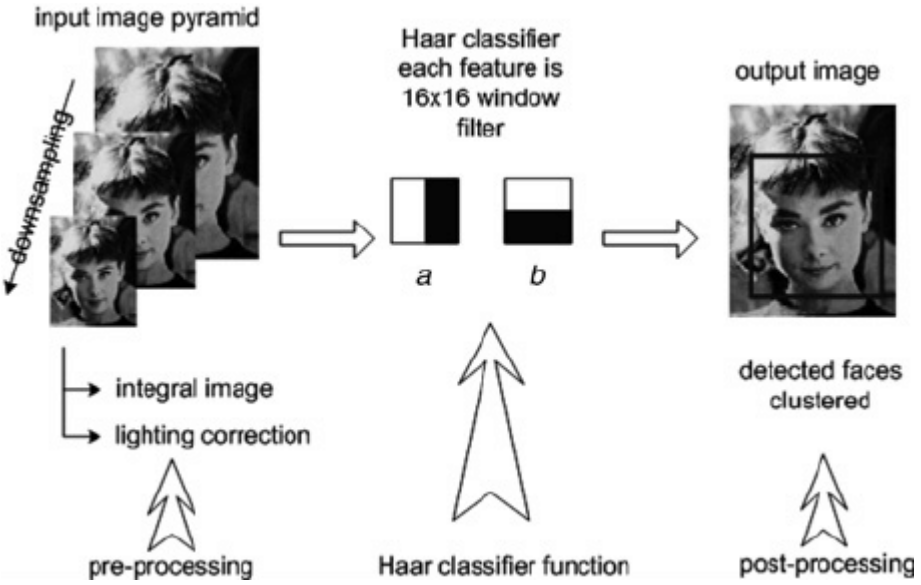


Fig 3.3 – Process of Haar Cascade Algorithm

Haar Cascade is a popular algorithm for object detection that uses a machine learning-based approach to identify objects in images or videos. In face detection, the Haar Cascade algorithm can be used to detect faces in an image or video frame. The algorithm works by using a set of pre-trained Haar features, which are simple rectangular filters that can detect edges, corners, and other features in an image. These Haar features are used to build a classifier that can distinguish between faces and non-faces. The Haar Cascade classifier is trained on a large dataset of images that contain faces and non-faces. During the training phase, the algorithm learns to recognize the patterns and features that are common to faces, such as the presence of eyes, nose, mouth, and the general shape of the face. After the training of the Haar Cascade classifier, it becomes capable of detecting faces in a new image or video frame. The detection process involves sliding a window across the image and utilizing Haar features on each window to identify the presence of a face. Once a window is identified as containing a face, the algorithm creates a bounding box

around the detected face. To improve the accuracy of face detection, multiple Haar Cascade classifiers can be applied in a cascaded fashion. In a cascaded classifier, the input image is first passed through a series of classifiers, each of which is designed to detect different facial features. The output of each classifier is used to refine the search space for the next classifier, resulting in faster and more accurate face detection. In summary, Haar Cascade is a popular algorithm for face detection that uses a machine learning-based approach to detect faces in images or videos. The algorithm works by using pre-trained Haar features to build a classifier that can distinguish between faces and non-faces, and applies a sliding window approach to detect faces in an image.

3.1.11 Weapon Detection and Notification alerts:

The YOLOv7 algorithm is a cutting-edge object identification algorithm that can accurately and instantly detect things. The algorithm was employed in this research to detect weapons. 714 pictures of various weapons from a proprietary dataset were used to train the algorithm. To assure the quality of the dataset, the images were preprocessed and the dataset was labelled using the free and open-source labelling application LabelImg.
A deep neural network is used by the YOLOv7 algorithm to identify objects in pictures and videos. To understand the characteristics of various objects, the algorithm is trained using a sizable collection of photos and videos. The programme can accurately identify the type of object and find several objects in a picture or video.

In this project, the YOLOv7 algorithm was fine-tuned using various techniques, including data augmentation and normalization. Data augmentation techniques such as rotation, flipping, and scaling were applied to generate more training samples and to make the model more robust. Data normalization techniques were also applied to ensure that the input data had a consistent range of values.

The model was trained to detect weapons in real-time and trigger notifications to the concerned authorities in case of a positive detection. The notification system was designed to send an immediate notification to the authorities. This approach helped to ensure that the authorities could respond quickly to potential threats and take appropriate action to prevent any harm to the public.

The YOLOv7 algorithm was chosen for its high accuracy and real-time detection capabilities. The algorithm can detect weapons in real-time, making it suitable for use in public spaces such as airports, public transportation, and public events. The algorithm can also be integrated with other security systems, such as CCTV cameras and metal detectors, to provide additional security measures.

In this study, the YOLOv7 algorithm was employed for weapon detection. The model was developed to identify weapons instantly and provide information to the appropriate authorities in the event of a positive find. The YOLOv7 algorithm is a cutting-edge object identification algorithm that can accurately and instantly detect things. The algorithm can be used to stop possible threats and guarantee public safety in a variety of settings, including airport security, public transportation, and public events.

3.1.11.1 WhatsApp Notification:

WhatsApp is a widely-used messaging platform with a key feature of sending messages, images, and media easily and quickly. It is an excellent tool for sending alerts and notifications in real-time.PyWhatKit is a library developed in Python that offers a platform for transmitting WhatsApp messages by integrating it with Python code. The “sendwhatmsg()” function within PyWhatKit takes in the recipient's phone number (in international format), the message, and the optional time for sending the message. This function can send messages to a single recipient or multiple recipients simultaneously. WhatsApp messages through PyWhatKit provide advantages over traditional SMS alerts as they can include media, such as images and videos, and can be sent over Wi-Fi or cellular data connections. This ensures that recipients can receive alerts even with a weak cellular signal. Overall, PyWhatKit makes it simple to integrate WhatsApp messaging into Python projects, allowing for the effective and timely delivery of alerts and notifications.

3.1.11.2 Email Notification:

Email has long been a widely adopted mode of communication, and its potential for use in sending alerts and notifications in various projects is immense. With the help of modern programming libraries, such as Python's "smtplib" library, sending emails via code has become a seamless process. The "smtplib" library may be used to build an email object including the sender's email address, the recipient's email

address, the subject line, and the body of the message in order to send email alerts in a project. The Simple Mail Transfer Protocol (SMTP) server may be used to send the email object to the recipient's email address after it has been formed. The convenience and ease of email-based communication through code have made it a preferred choice for notifications and alerts in numerous projects. The utilization of such libraries not only facilitates the process of sending emails, but also offers the flexibility to automate and customize email alerts based on specific project requirements. Overall, using email alerts can be an effective way to keep users informed and up-to-date in a project. With the help of modern programming libraries, sending email alerts through code has become a streamlined process that can enhance the functionality of any project.

3.1.11.3 Sound Alert:

Computer vision technology has been increasingly used for real-time object detection and identification, especially in security and surveillance applications. Detecting weapons on screen is a particularly important application of this technology. When a weapon is detected on screen, it is important to alert the relevant personnel quickly and effectively. In this project, an alarm sound is used to notify the user when a weapon is detected on screen. Using sound alerts to notify users of the presence of a weapon has several advantages, including its effectiveness, versatility, and customizability. However, the use of sound alerts should be carefully considered to avoid being disruptive or distracting. The volume and frequency of the alarm sound should be appropriately set to ensure that it is both effective and appropriate for the situation. Overall, the use of alarm sounds in this project can be a valuable tool for detecting weapons on screen and enhancing the safety and security of various applications.

3.1.12 Implementing UI:

The Streamlit framework was used to implement the user interface for this project. The UI was designed to display the live video feed and the results of the object detection and recognition processes. The UI was user-friendly and included various controls such as start, stop, and reset buttons. The UI also included a log section that displayed the notifications sent via email.

Streamlit:

Streamlit is a Python library that simplifies the process of creating interactive web applications for data science and machine learning projects. It provides an easy-to-use interface for building custom web applications without requiring knowledge of HTML, CSS, or JavaScript code. The library is designed to make it easy for developers to get started and build applications quickly. It integrates with popular data science libraries like Pandas, NumPy, and scikit-learn, making it easy to incorporate these tools into your application. One of the significant advantages of Streamlit is its ability to update in real-time, allowing users to see changes in their data visualizations immediately. Streamlit is becoming increasingly popular among data scientists and machine learning engineers because of its ease of use and the variety of built-in widgets that make it easy to create interactive visualizations. Overall, Streamlit is an excellent choice for anyone looking to create web applications for data science or machine learning projects quickly and efficiently.

3.2 Model Evaluation:

Object detection models are evaluated using test sets and three primary metrics: Accuracy, Precision, and Recall. Accuracy measures the overall correctness of the model's predictions by computing the proportion of correct predictions to the total number of predictions. The range of Accuracy is between 0 and 1, where a score of 1 represents perfect accuracy, and a score of 0 represents no accuracy.

$$\text{Accuracy} = (\text{Number of Correct Predictions}) / (\text{Total Number of Predictions})$$

Precision is the ratio of true positives to the total number of positive predictions made by the model. It represents the proportion of positive predictions that are actually true positives. A high precision score indicates that the model generates a minimal number of false positive predictions.

$$\text{Precision} = (\text{True Positives}) / (\text{True Positives} + \text{False Positives})$$

Recall is the ratio of true positives to the total number of actual positive samples in the dataset. It represents the proportion of actual positive samples that the model correctly identifies as positive. A high recall score indicates that the model can successfully identify the majority of the positive samples in the dataset.

$$\text{Recall} = (\text{True Positives}) / (\text{True Positives} + \text{False Negatives})$$

3.3 Data flow diagram:

To illustrate the movement of information throughout a procedure or system, one might use a Data-Flow Diagram (DFD). A data-flow diagram does not include any decision rules or loops, as the flow of information is entirely one-way. A flowchart can be used to illustrate the steps used to accomplish a certain data-driven task. Several different notations exist for representing data-flow graphs. Each data flow must have a process that acts as either the source or the target of the information exchange. Rather than utilizing a data-flow diagram, users of UML often substitute an activity diagram. In the realm of data-flow plans, site-oriented data-flow plans are a subset. Identical nodes in a data-flow diagram and a Petri net can be thought of as inverted counterparts since the semantics of data memory are represented by the locations in the network. Structured data modeling (DFM) includes processes, flows, storage, and terminators.

The whole system is shown as a single process in a level DFD. Each step in the system's assembly process, including all intermediate steps, are recorded here. The "basic system model" consists of this and 2-level data flow diagrams.

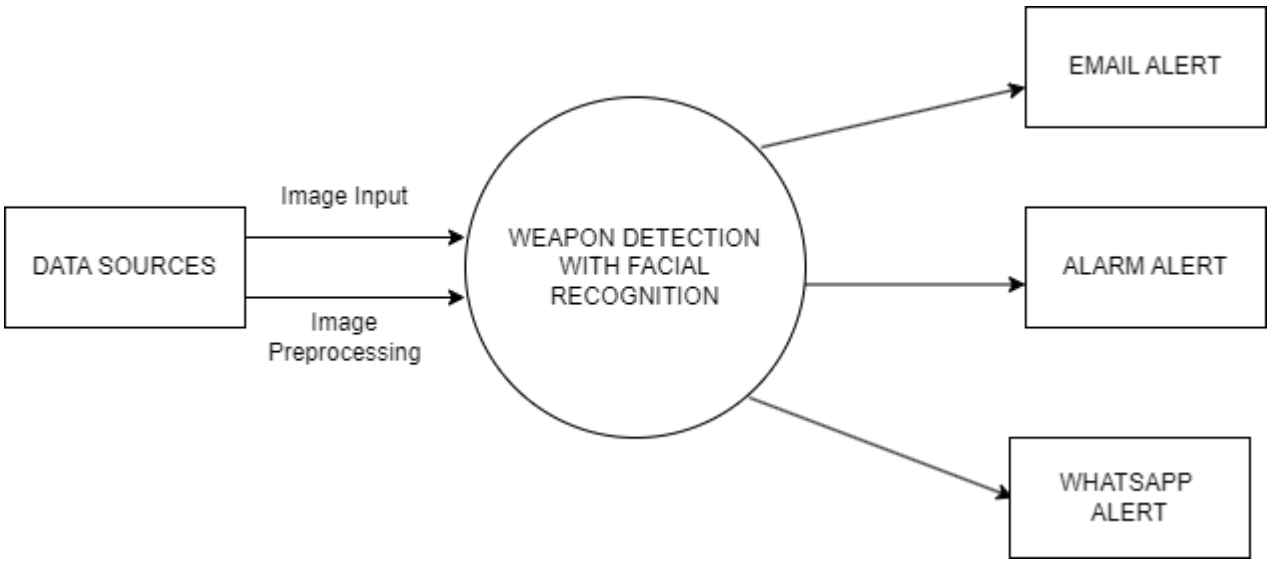


Fig 3.4 – Data Flow Diagram Level 0

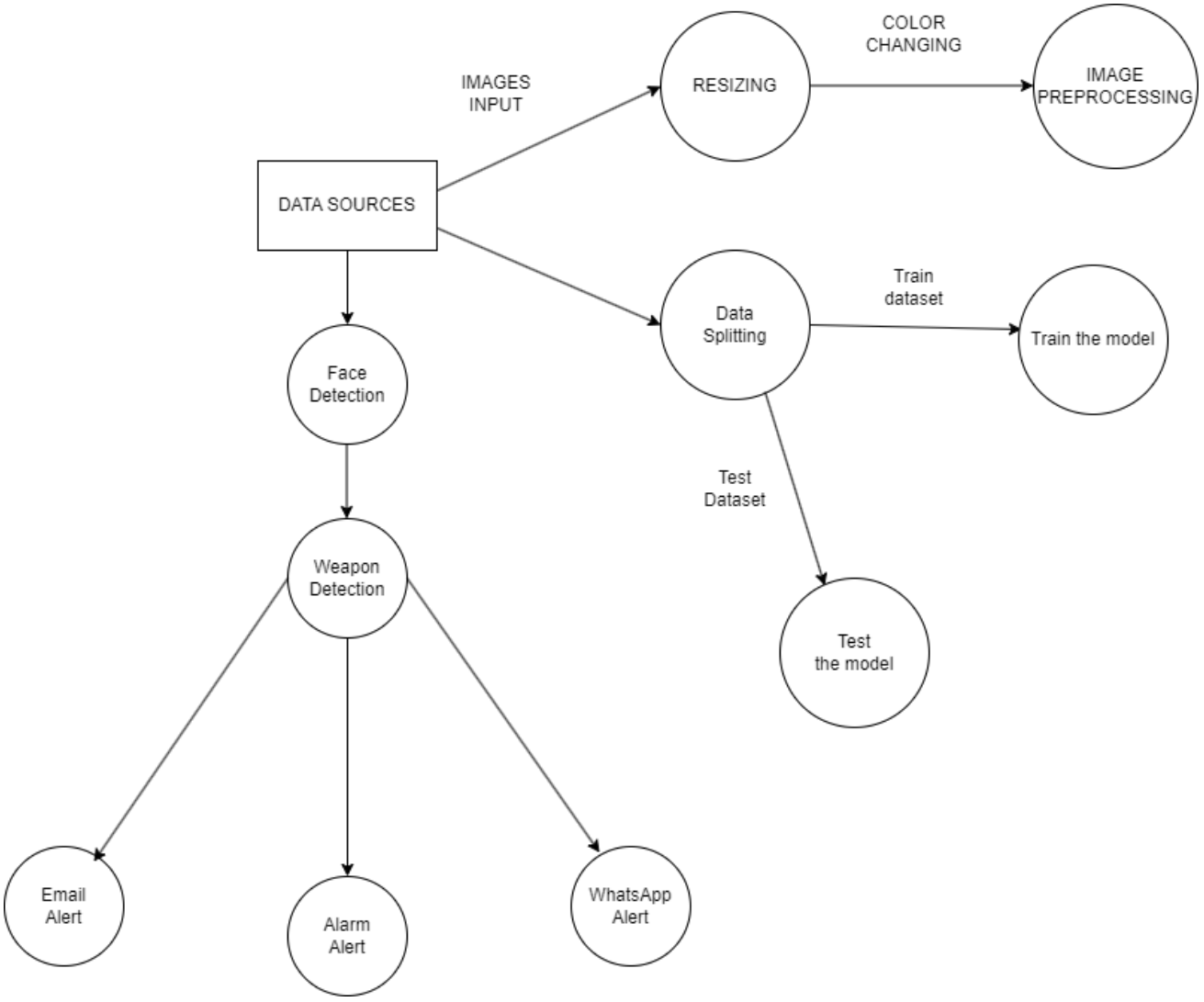


Fig 3.5 – Data Flow Diagram Level 1

3.4 Use-Case Diagram:

The possible interactions between the user, the dataset, and the algorithm are often depicted in a use case diagram. It's created at the start of the procedure.

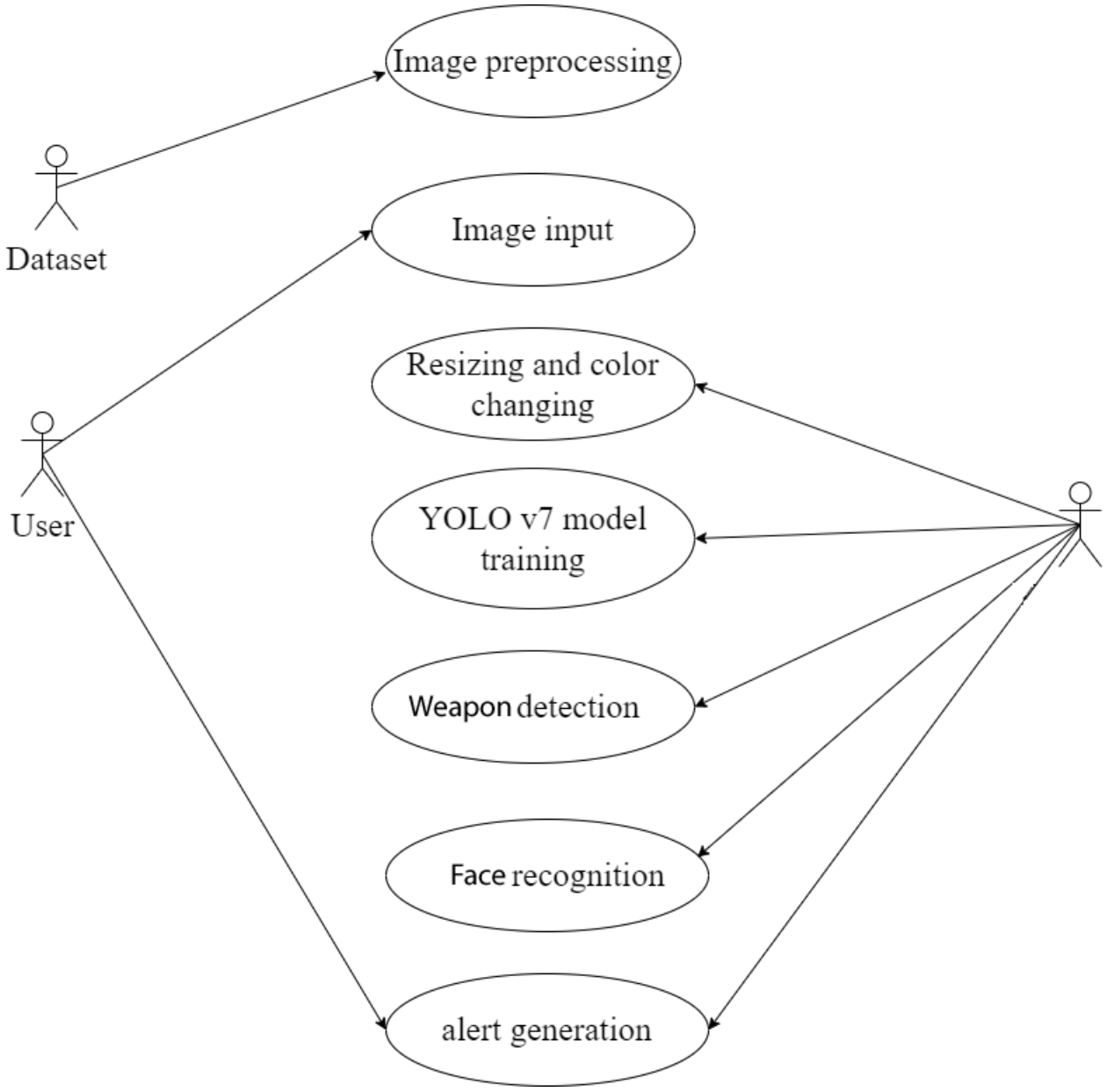


Fig 3.6 – Use-Case Diagram

3.5 Activity Diagram:

An activity diagram, in its most basic form, is a visual representation of the sequence in which tasks are performed. It depicts the sequence of operations that make up the overall procedure. They are not quite flowcharts, but they serve a comparable purpose.

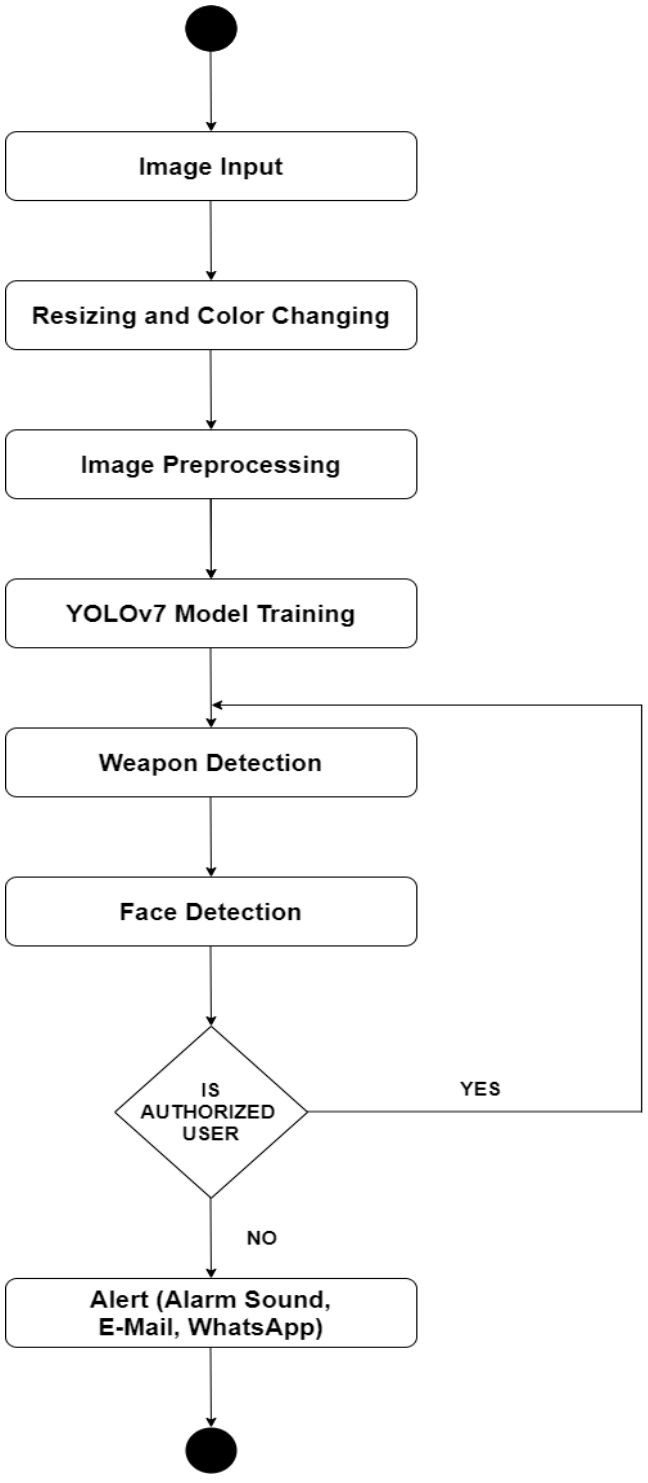


Fig 3.7 – Activity Diagram

3.6. Sequence Diagram:

These are another type of interaction-based diagram used to display the workings of the system. They record the conditions under which objects and processes cooperate.

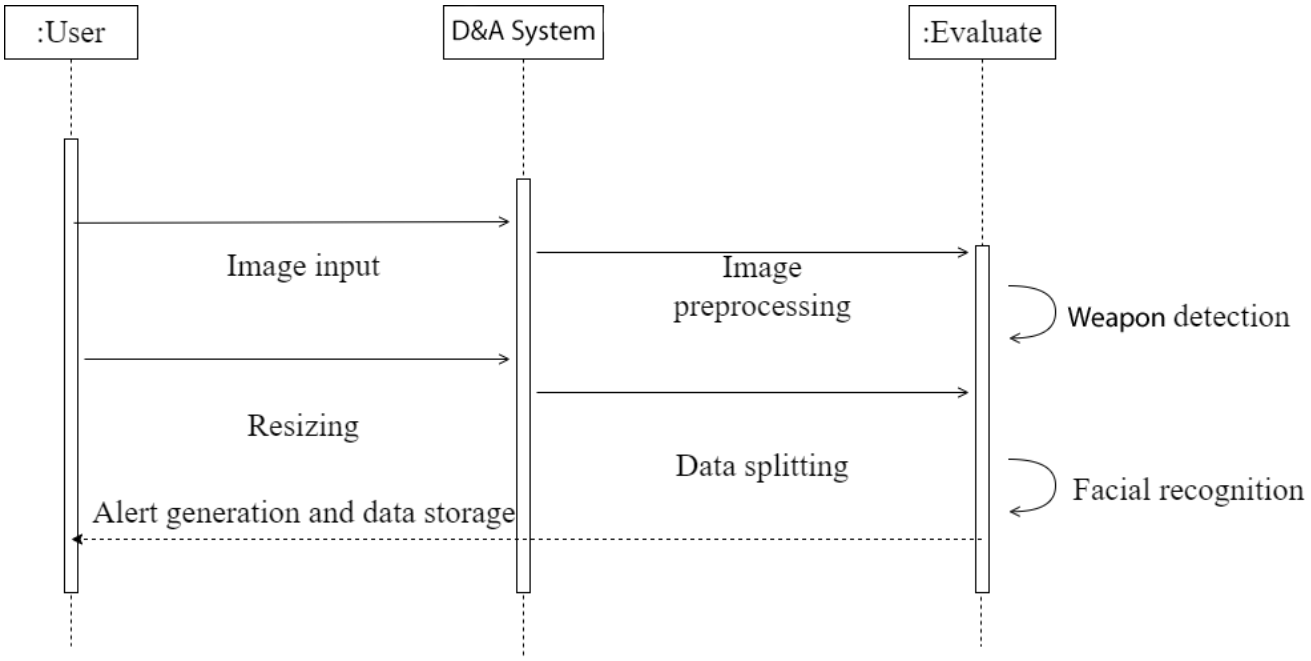


Fig 3.8 – Sequence Diagram

3.7 CLASS DIAGRAM

In essence, this is a "context diagram," another name for a contextual diagram. It simply stands for the very highest point, the 0 Level, of the procedure. As a whole, the system is shown as a single process, and the connection to externalities is shown in an abstract manner.

- A + indicates a publicly accessible characteristic or action.
- A - a privately accessible one.
- A # a protected one.
- A - denotes private attributes or operations.

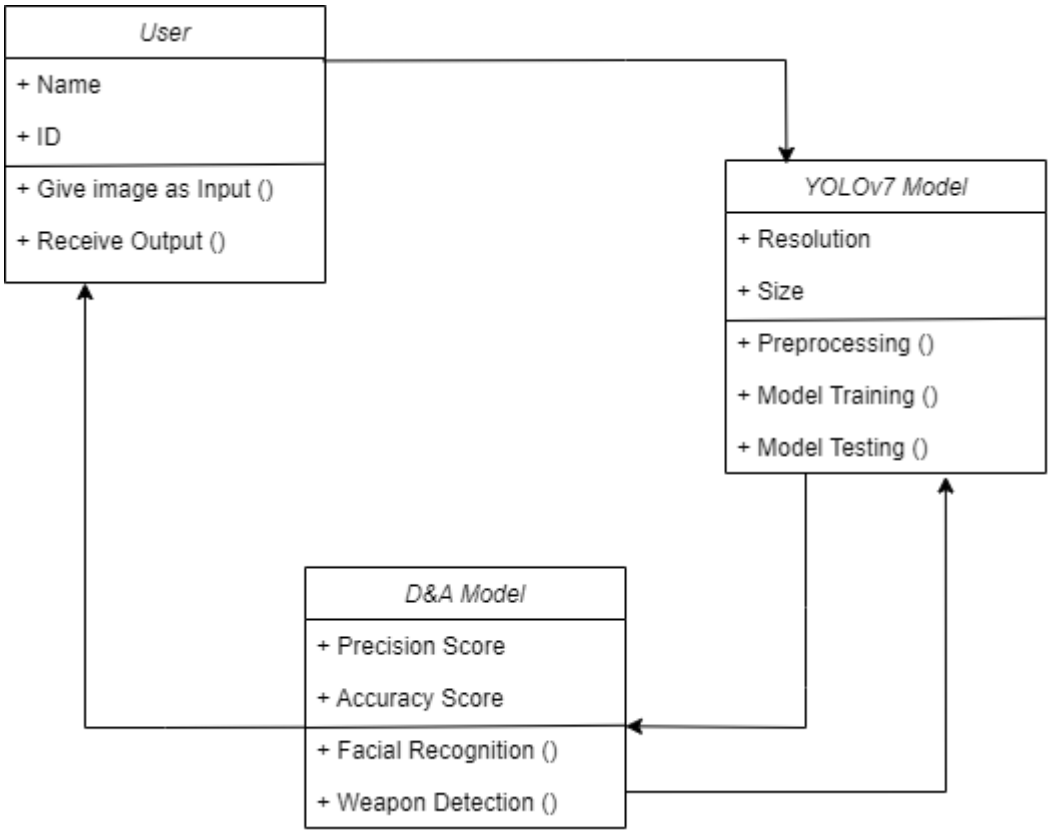


Fig 3.9 – Class Diagram

CHAPTER 4

METHODOLOGY

4.1 MODULE 1: Data Collection and Preprocessing:

Data collection and preprocessing for weapon detection using face recognition with YOLOv7 involve several steps, including gathering and labeling images, face detection and recognition, and dataset preprocessing.

The first step in data collection for weapon detection using face recognition with YOLOv7 involves searching for images online or capturing them using cameras or drones. The images should be diverse in terms of backgrounds, lighting conditions, and types of weapons. In addition to labeling the weapons using bounding boxes, the faces in the images should be labeled with the corresponding identity of the person.

To label the faces in the images, a facial recognition algorithm can be used. The algorithm should be trained on a separate dataset of faces to recognize the faces of individuals in the images. The facial recognition algorithm should detect the location of the faces in the images and output the corresponding identity of the person.

Once the dataset is collected and labeled, it is important to preprocess it to ensure that the images are standardized and ready for training. Preprocessing steps typically include resizing the images to a fixed size, converting them to the YOLO format, and augmenting them to increase the diversity of the dataset. Augmentation techniques such as flipping, rotating, and changing brightness can help prevent overfitting and improve model generalization.

The preprocessing pipeline for weapon detection using face recognition with YOLOv7 also involves combining the face and weapon labels to create a unified dataset for training. This can be done by associating the face label with the corresponding weapon label in each image.

To improve the accuracy of the facial recognition algorithm, it is also important to preprocess the face images. This can involve cropping and resizing the faces to a

fixed size, normalizing the pixel values, and performing data augmentation techniques such as rotating or flipping the images.

After the dataset is preprocessed, it is ready for training the YOLOv7 model. During training, the YOLOv7 algorithm learns to detect weapons and recognize faces simultaneously. In order to minimize the loss function during training, the model's weights and biases are adjusted by repeatedly iterating over the dataset.

4.2 MODULE 2: Model Training:

Training a YOLOv7 model for weapon detection using face recognition involves several steps, including setting up the training environment, configuring the YOLOv7 model, and running the training process.

Firstly, the training environment should be set up with the necessary software and hardware. YOLOv7 requires a powerful GPU to train on, such as a NVIDIA GTX 1080Ti or higher. Additionally, software such as Python, PyTorch, and CUDA should be installed, along with any necessary libraries such as OpenCV and NumPy.

Next, the YOLOv7 model should then be set up specifically for the job of utilising facial recognition to find weapons. This entails picking a YOLOv7 model that has already been trained and customising it for the relevant dataset and detection job. The number of classes (in this example, the number of weapons to identify), the size of the input photos, and other hyperparameters like the learning rate and batch size should all be set for the model.

During the training process, the YOLOv7 model should be trained to detect potential threats based on both the detected weapon and the identity of the person holding the weapon. This can be achieved through multi-task learning or feature fusion, which involves combining the object detection and facial recognition features of the model. The training process for YOLOv7 involves iterating over the dataset multiple times, with each iteration referred to as an epoch. During each epoch, the YOLOv7 algorithm processes the images in batches, computes the loss function for each batch, and updates the model's weights and biases based on the gradient of the loss function. The loss function measures the difference between the predicted and actual bounding boxes and facial recognition outputs for each

object in the image and penalizes the model for false positives and false negatives.

After training is finished, the model may be tested against a validation dataset to see how accurate and precise it is. If the model works well, it may be used in real-world circumstances to find firearms and identify faces. To make sure that the model is accurate and up to date with new kinds of weapons and faces, it is crucial to regularly retrain it and to continually analyze its performance.

4.3 MODULE 3: Prediction of Output

The prediction output of a trained YOLOv7 model for weapon detection using face recognition involves providing information about the location and size of detected weapons, as well as the identity of the person holding the weapon. The output can be in the form of bounding boxes around the detected weapons, along with the corresponding identity of the person holding the weapon.

The YOLOv7 model would take an input image or video stream and process it through a series of CNNs to detect potential threats, using both facial recognition and object detection techniques. The model would first detect the faces in the image and identify the corresponding identity of the person. The model would then detect the weapons in the image and output bounding boxes around the detected weapons.

The output would also include associated confidence scores for both the detected weapons and the facial recognition outputs. The confidence scores indicate the YOLOv7 model's degree of certainty that the detected object is a weapon or the detected face corresponds to a particular individual.

By setting a threshold for the confidence score, the model can be tuned to detect only highly confident predictions, reducing false positives and improving accuracy. Additionally, the output of the YOLOv7 model can be further processed and analyzed, for example, by triggering an alarm or notifying security personnel when the confidence score exceeds the threshold and a potential object of interest is detected, enabling a prompt response to potential security threats or anomalies in real-time. Such post-processing techniques can enhance the effectiveness of the YOLOv7 model in various applications, including surveillance, object recognition, and autonomous driving, among others

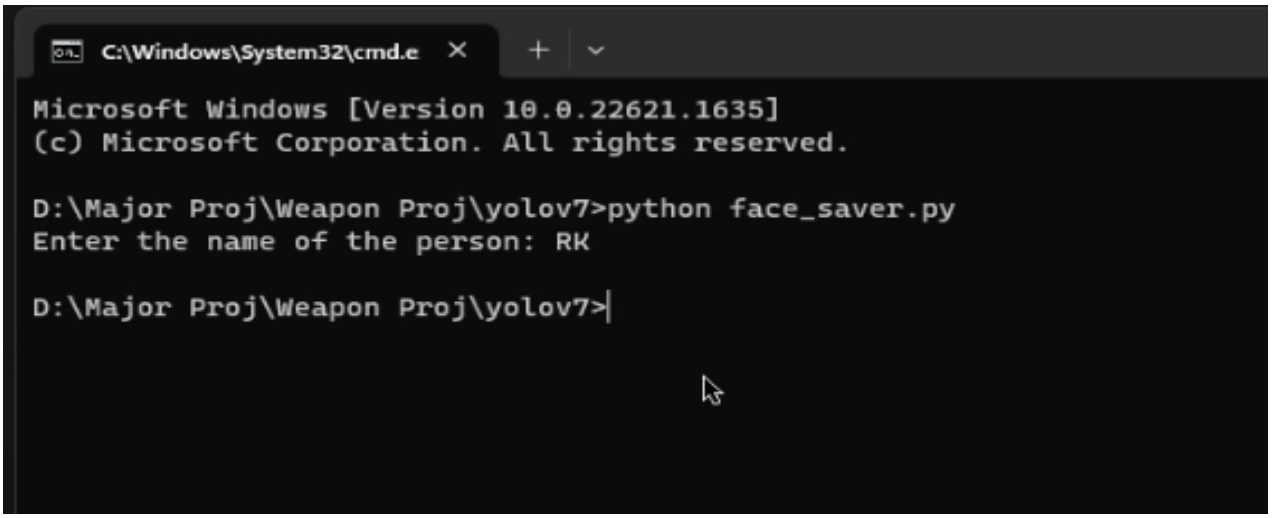
CHAPTER 5

CODING AND TESTING

5.1 Face Saver code:

```
import cv2
import os

# Create a cascade classifier for face detection
face_cascade = cv2.CascadeClassifier( cv2.data.haarcascades +
'haarcascade_frontalface_alt.xml')

# Create a VideoCapture object to capture frames from the webcam
cap = cv2.VideoCapture(0)

# Get the name of the person whose face is being registered
name = input("Enter the name of the person: ")

# Create a folder with the name of the person (if it doesn't exist already)
if not os.path.exists('fdata/'+name):
    os.makedirs('fdata/'+name)

# Initialize the counter for the number of faces registered
count = 0

# Keep capturing frames from the webcam and registering faces until the user presses 'q'
while True:
    # Read a frame from the webcam
    ret, frame = cap.read()

    # Convert the frame to grayscale for faster processing
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect faces in the grayscale frame
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    # Draw rectangles around the detected faces and save them to the folder
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]
        file_path = os.path.join('fdata/'+name, f"{name}_{count}.jpg")
        cv2.imwrite(file_path, roi_color)
        count += 1
        if count==100:
            break

    # Display the frame with rectangles around the detected faces
    cv2.imshow('frame', frame)
```

```
    # Break out of the loop if the user presses 'q'
    if cv2.waitKey(1) & 0xFF == ord('q')or count==100:
        break

# Release the VideoCapture object and close all windows
cap.release()
cv2.destroyAllWindows()
```



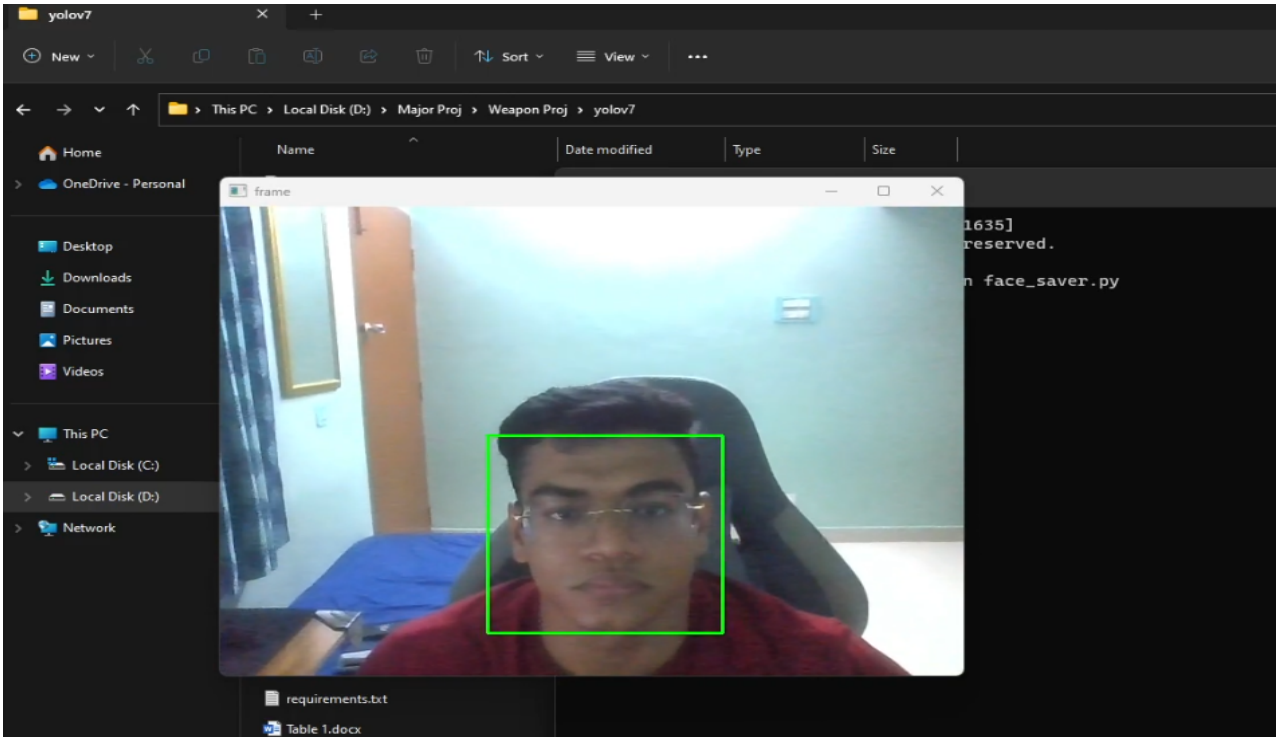


Fig: 5.1 Registering the Person's Face for Authorisation

5.2 Face Detector code:

```
import cv2
import os
import numpy as np

import datetime
import smtplib
import subprocess

# Email login credentials
email = "weapondetector@gmail.com"
password = "ehycnaltfqdgqkdo"

# Email details
to_email = "sg7744@srmist.edu.in"
subject = "ALERT"
body = "Unauthorized Access has been detected!"

# Create an SMTP object
smtp_server = smtplib.SMTP("smtp.gmail.com", 587)
smtp_server.starttls()

# Login to the email account
smtp_server.login(email, password)

# Create the email message
message = f"Subject: {subject}\n\n{body}"

# Create a cascade classifier for face detection
face_cascade = cv2.CascadeClassifier( cv2.data.haarcascades +
'haarcascade_frontalface_alt.xml')

# Create a recognizer for face recognition
recognizer = cv2.face.LBPHFaceRecognizer_create()

# Get the paths of the folders containing the training data
data_path = 'fdata/'
folders = os.listdir(data_path)

# Initialize dictionaries for mapping between person names and labels
name_to_label = {}
label_to_name = {}

# Initialize lists for storing the training data and labels
X_train = []
y_train = []
```

```
# Assign unique integer labels to each person in the dataset
label = 0
for folder in folders:
    name_to_label[folder] = label
    label_to_name[label] = folder
    files = os.listdir(data_path + folder)
    for file in files:
        img = cv2.imread(data_path + folder + '/' + file)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for (x, y, w, h) in faces:
            roi_gray = gray[y:y+h, x:x+w]
            X_train.append(roi_gray)
            y_train.append(label)
    label += 1

# Train the recognizer on the training data and labels
recognizer.train(X_train, np.array(y_train))

# Create a VideoCapture object to capture frames from the webcam
cap = cv2.VideoCapture(0)

# Keep detecting faces and recognizing them until the user presses 'q'
while True:
    # Read a frame from the webcam
    ret, frame = cap.read()

    # Convert the frame to grayscale for faster processing
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect faces in the grayscale frame
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    name ="no face"
    # Recognize the detected faces and display their names
    for (x, y, w, h) in faces:
        roi_gray = gray[y:y+h, x:x+w]
        label, confidence = recognizer.predict(roi_gray)
        print(confidence)
        if confidence<65:
            name = label_to_name[label]
        else:
            name="Unauthorized"



        cv2.putText(frame, name, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255,
0), 2)
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

```
# Display the frame with names and rectangles around the detected faces
cv2.imshow('frame', frame)

# Break out of the loop if the user presses 'q'
if cv2.waitKey(1) & 0xFF == ord('q')or name=="Unauthorized":


    break

# Release the VideoCapture object and close all windows
cap.release()
cv2.destroyAllWindows()
subprocess.run("python detect.py --weights best.pt --conf 0.5 --img-size 640 --source 0
--view-img --no-trace")

smtp_server.sendmail(email, to_email, message)
smtp_server.quit()
```

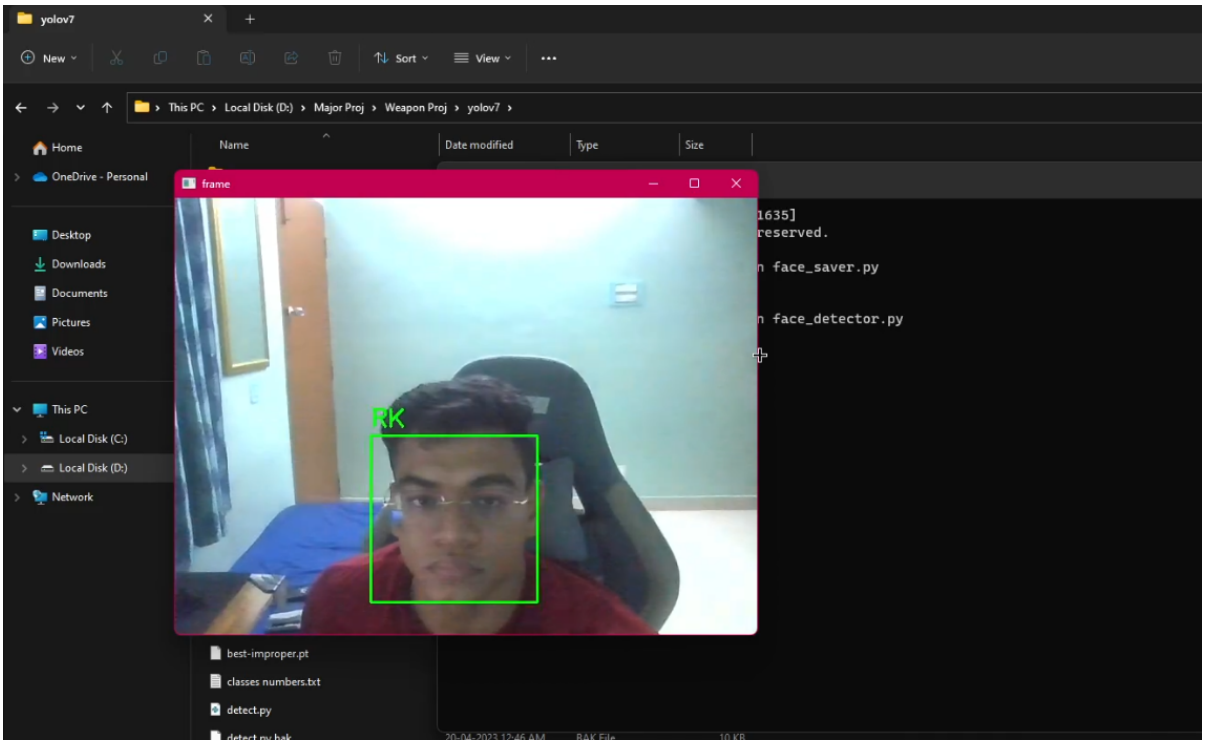
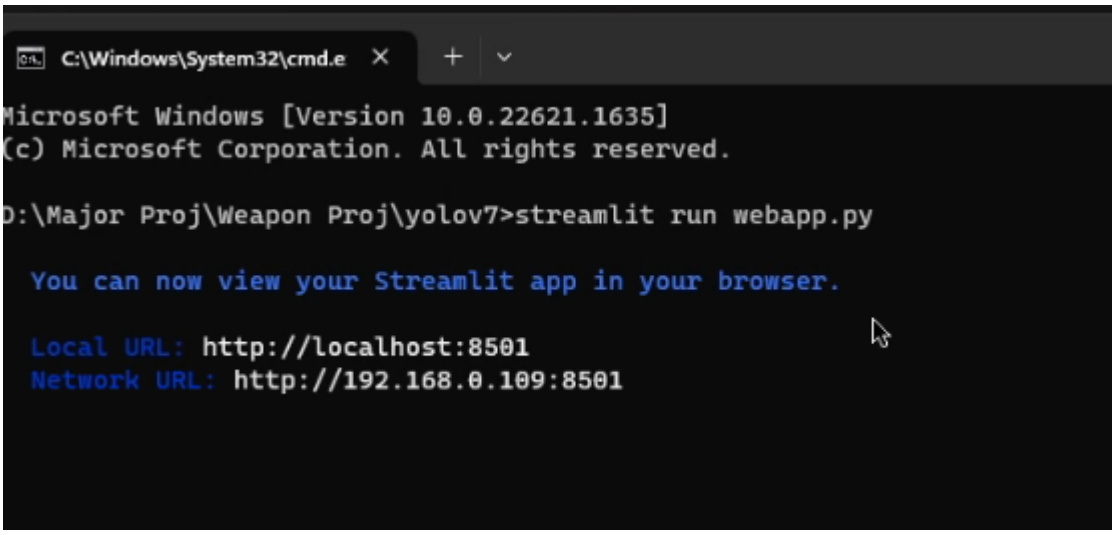


Fig: 5.2 Recognising the Person's face

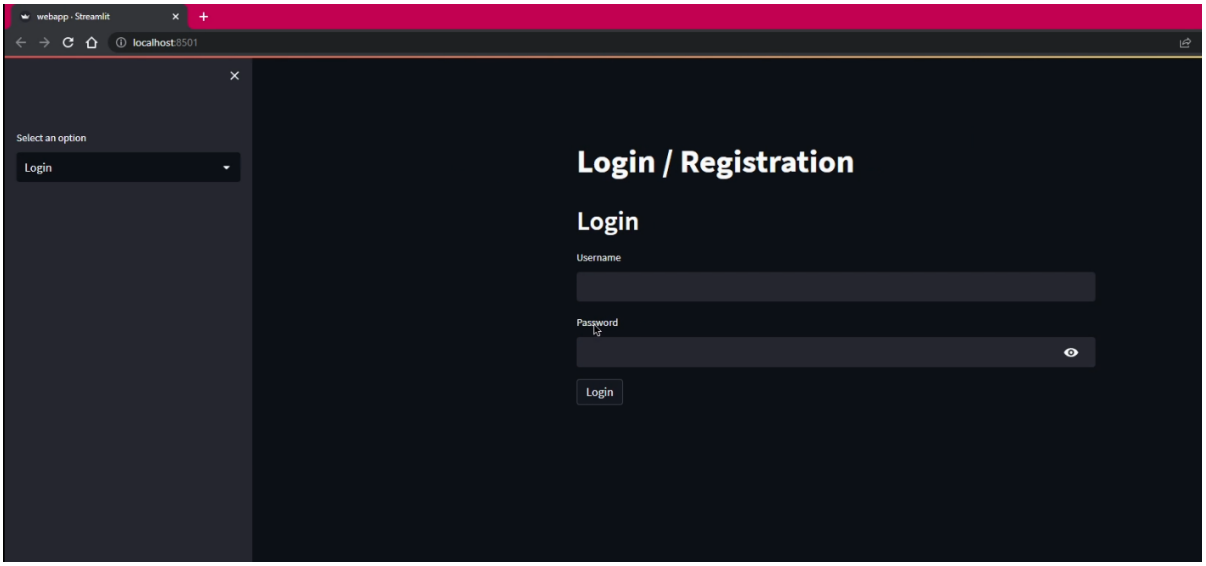
5.3: Main Page code:

```
import streamlit as st
import pandas as pd
import subprocess
import os

# Define a function to register a new user
def register(username, password):
    # Load the existing user data (or create an empty dataframe if it doesn't exist)
    try:
        user_data = pd.read_csv("user_data.csv")
    except:
        user_data = pd.DataFrame(columns=["username", "password"])

    # Check if the username already exists
    if username in user_data["username"].values:
        st.error("Username already exists.")
    else:
        # Add the new user to the user data
        new_user = {"username": username, "password": password}
        user_data = user_data.append(new_user, ignore_index=True)
        user_data.to_csv("user_data.csv", index=False)
        st.success("Registration successful.")

# Define a function to log in an existing user
def login(username, password):
    # Load the user data
    try:
        user_data = pd.read_csv("user_data.csv")
    except:
        st.error("No users registered yet.")
        return
    # Check if the username and password match
    if (username in user_data["username"].values) & (password in
user_data["password"].values):
        st.success("Login successful.")
        subprocess.run("python face_detector.py")
    else:
        st.error("Incorrect username or password.")

# Define the Streamlit app
def app():
    st.title("Login / Registration")

    # Create a sidebar with a menu of options
    menu = ["Login", "Register"]
```

```
    choice = st.sidebar.selectbox("Select an option", menu)

    # Show the appropriate form based on the user's menu choice
    if choice == "Login":
        st.header("Login")
        username = st.text_input("Username")
        password = st.text_input("Password", type="password")
        if st.button("Login"):
            login(username, password)
    elif choice == "Register":
        st.header("Register")
        username = st.text_input("Username")
        password = st.text_input("Password", type="password")
        confirm_password = st.text_input("Confirm Password", type="password")
        if st.button("Register"):
            if password == confirm_password:
                register(username, password)
            else:
                st.error("Passwords do not match.")
app()
```



5.4. User Login and Verification:



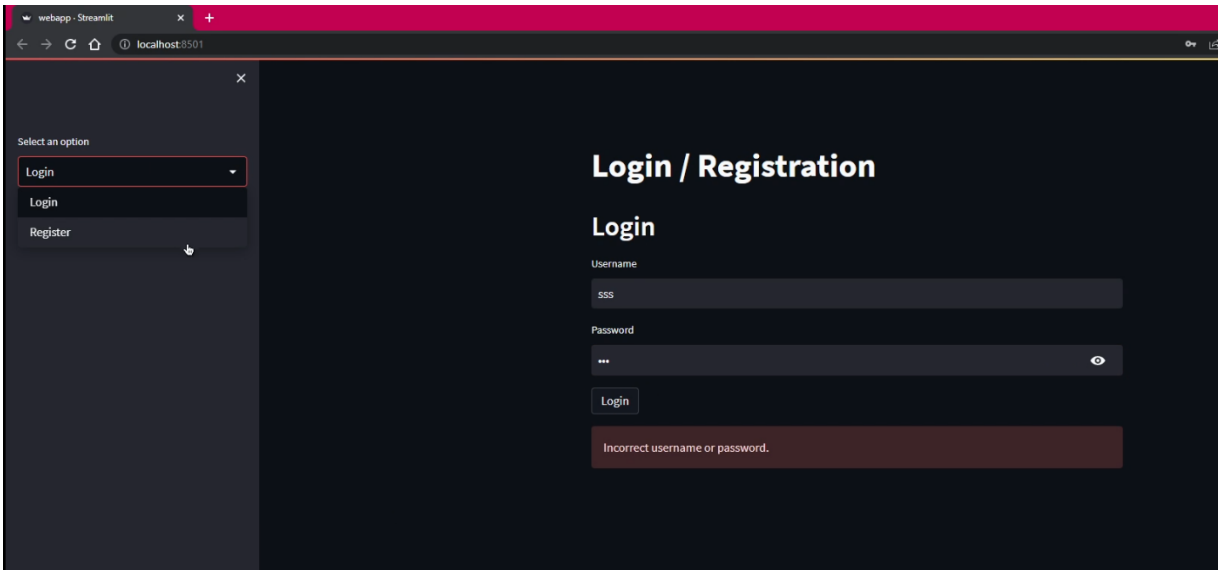
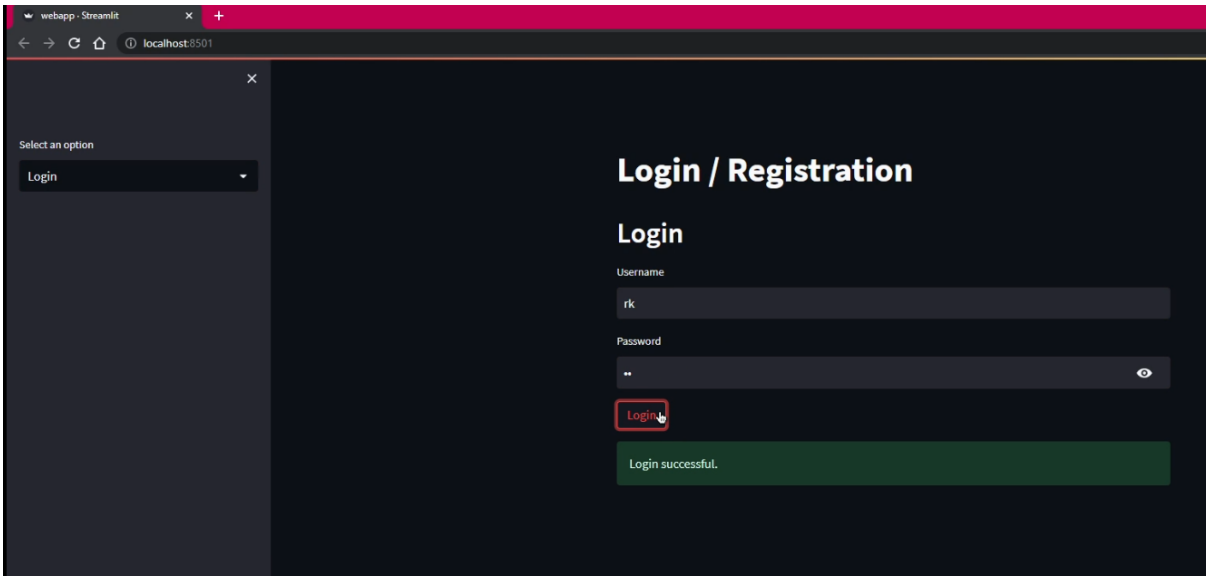


Fig: 5.3 Unauthorized user login



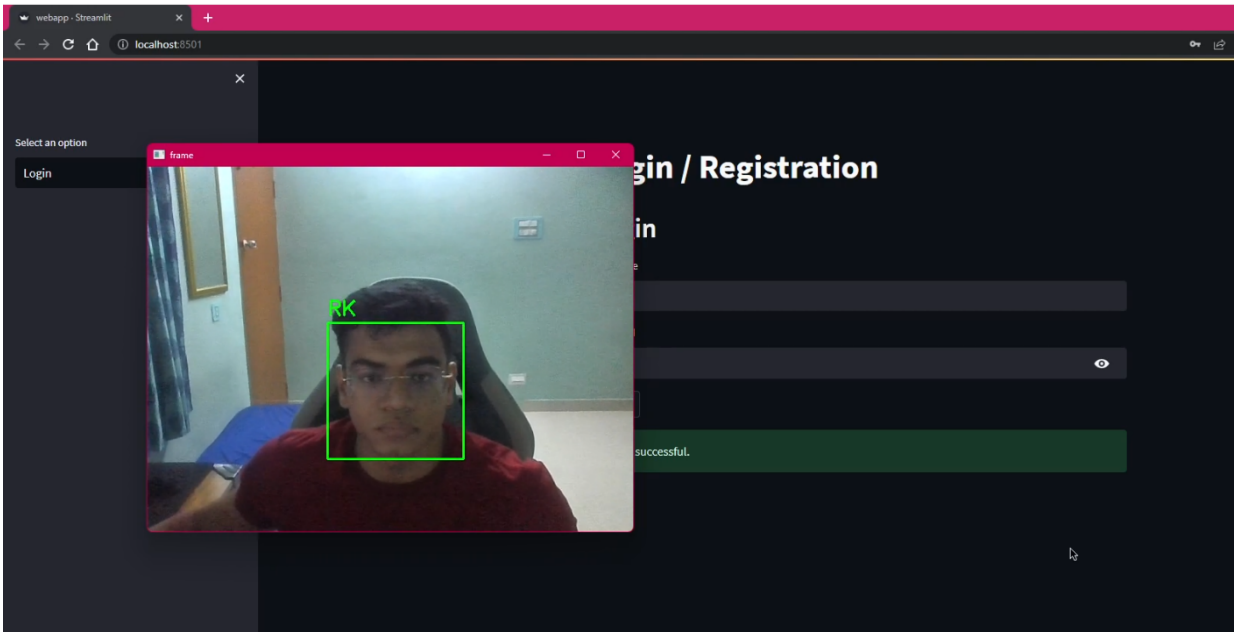


Fig: 5.4 Authorized user login

5.4: Weapon Detector code:

```
import argparse
import time
from pathlib import Path
import pywhatkit as pw
import datetime
import simpleaudio as sa
import cv2
import torch
import torch.backends.cudnn as cudnn
from numpy import random

from models.experimental import attempt_load
from utils.datasets import LoadStreams, LoadImages
from utils.general import check_img_size, check_requirements, check_imshow,
non_max_suppression, apply_classifier, \
    scale_coords, xyxy2xywh, strip_optimizer, set_logging, increment_path
from utils.plots import plot_one_box
from utils.torch_utils import select_device, load_classifier, time_synchronized,
TracedModel



def detect(save_img=False):
    source, weights, view_img, save_txt, imgsz, trace = opt.source, opt.weights,
opt.view_img, opt.save_txt, opt.img_size, not opt.no_trace
    save_img = not opt.nosave and not source.endswith('.txt')  # save inference images
    webcam = source.isnumeric() or source.endswith('.txt') or source.lower().startswith(
        ('rtsp://', 'rtmp://', 'http://', 'https://'))

    # Directories
    save_dir = Path(increment_path(Path(opt.project) / opt.name, exist_ok=opt.exist_ok))
# increment run
    (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  #
make dir

    # Initialize
    set_logging()
    device = select_device(opt.device)
    half = device.type != 'cpu'  # half precision only supported on CUDA

    # Load model
    model = attempt_load(weights, map_location=device)  # load FP32 model
    stride = int(model.stride.max())  # model stride
    imgsz = check_img_size(imgsz, s=stride)  # check img_size
```

```
  if trace:
      model = TracedModel(model, device, opt.img_size)
   if half:
      model.half()  # to FP16
      # Second-stage classifier
   classify = False
   if classify:
      modelc = load_classifier(name='resnet101', n=2)  # initialize
      modelc.load_state_dict(torch.load('weights/resnet101.pt',
map_location=device)['model']).to(device).eval()
   # Set Dataloader
   vid_path, vid_writer = None, None
   if webcam:
      view_img = check_imshow()
      cudnn.benchmark = True  # set True to speed up constant image size inference
      dataset = LoadStreams(source, img_size=imgsz, stride=stride)
   else:
      dataset = LoadImages(source, img_size=imgsz, stride=stride)

   # Get names and colors
   names =
["Weapon","Weapon","Weapon","Weapon","Weapon","Weapon","Weapon","Weapo
n","Weapon"]
   colors = [[random.randint(0, 255) for _ in range(3)] for _ in names]

   # Run inference
   if device.type != 'cpu':
      model(torch.zeros(1, 3, imgsz, imgsz).to(device).type_as(next(model.parameters())))
# run once
   old_img_w = old_img_h = imgsz
   old_img_b = 1

   t0 = time.time()
   for path, img, im0s, vid_cap in dataset:
      img = torch.from_numpy(img).to(device)
      img = img.half() if half else img.float()  # uint8 to fp16/32
      img /= 255.0  # 0 - 255 to 0.0 - 1.0
      if img.ndimension() == 3:
         img = img.unsqueeze(0)

      # Warmup
      if device.type != 'cpu' and (old_img_b != img.shape[0] or old_img_h != img.shape[2]
or old_img_w != img.shape[3]):
         old_img_b = img.shape[0]
         old_img_h = img.shape[2]
         old_img_w = img.shape[3]
         for i in range(3):
            model(img, augment=opt.augment)[0]
```

```
        # Inference
        t1 = time_synchronized()
        with torch.no_grad():   # Calculating gradients would cause a GPU memory leak
            pred = model(img, augment=opt.augment)[0]
        t2 = time_synchronized()

        # Apply NMS
        pred = non_max_suppression(pred, opt.conf_thres, opt.iou_thres,
classes=opt.classes, agnostic=opt.agnostic_nms)
        t3 = time_synchronized()
        import winsound
        # Apply Classifier
        if classify:


# Play the audio file

            pred = apply_classifier(pred, modelc, img, im0s)

        # Process detections
        for i, det in enumerate(pred):  # detections per image
            if webcam:  # batch_size >= 1
                p, s, im0, frame = path[i], '%g: ' % i, im0s[i].copy(), dataset.count
            else:
                p, s, im0, frame = path, '', im0s, getattr(dataset, 'frame', 0)

            p = Path(p)  # to Path
            save_path = str(save_dir / p.name)  # img.jpg
            txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image' else
f'_{frame}')  # img.txt
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
            if len(det):
                # Rescale boxes from img_size to im0 size
                det[:, :4] = scale_coords(img.shape[2:], det[:, :4], im0.shape).round()

                # Print results
                for c in det[:, -1].unique():
                    n = (det[:, -1] == c).sum()  # detections per class
                    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string

                # Write results
                for *xyxy, conf, cls in reversed(det):
                    if save_txt:  # Write to file
                        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()  #
normalized xywh
                        line = (cls, *xywh, conf) if opt.save_conf else (cls, *xywh)  # label format
                        with open(txt_path + '.txt', 'a') as f:
                            f.write(('%g ' * len(line)).rstrip() % line + '\n')
```

```
                if save_img or view_img:  # Add bbox to image
                    wave_obj = sa.WaveObject.from_wave_file("alarm.wav")
                    # Play the audio file
                    play_obj = wave_obj.play()
                    now = datetime.datetime.now()
                    pw.sendwhatmsg("+919566178893", "Unauthorized weapon usage has
been detected!", now.hour, now.minute + 2)
                    label = f'{names[int(cls)]} {conf:.2f}'
                    plot_one_box(xyxy, im0, label=label, color=colors[int(cls)],
line_thickness=1)

            # Print time (inference + NMS)
            print(f'{s}Done. ({(1E3 * (t2 - t1)):.1f}ms) Inference, ({(1E3 * (t3 - t2)):.1f}ms)
NMS')

            # Stream results
            if view_img:
                cv2.imshow(str(p), im0)
                cv2.waitKey(1)  # 1 millisecond

            # Save results (image with detections)
            if save_img:
                if dataset.mode == 'image':
                    cv2.imwrite(save_path, im0)
                    print(f" The image with the result is saved in: {save_path}")
                else:  # 'video' or 'stream'
                    if vid_path != save_path:  # new video
                        vid_path = save_path
                        if isinstance(vid_writer, cv2.VideoWriter):
                            vid_writer.release()  # release previous video writer
                        if vid_cap:  # video
                            fps = vid_cap.get(cv2.CAP_PROP_FPS)
                            w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                            h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                        else:  # stream
                            fps, w, h = 30, im0.shape[1], im0.shape[0]
                            save_path += '.mp4'
                        vid_writer = cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
                    vid_writer.write(im0)
    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}" if
save_txt else ''
        #print(f"Results saved to {save_dir}{s}")

    print(f'Done. ({time.time() - t0:.3f}s)')
```

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default='yolov7.pt',
help='model.pt path(s)')
    parser.add_argument('--source', type=str, default='inference/images', help='source')  #
file/folder, 0 for webcam
    parser.add_argument('--img-size', type=int, default=640, help='inference size (pixels)')
    parser.add_argument('--conf-thres', type=float, default=0.25, help='object confidence
threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='IOU threshold for
NMS')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='display results')
    parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
    parser.add_argument('--save-conf', action='store_true', help='save confidences in
--save-txt labels')
    parser.add_argument('--nosave', action='store_true', help='do not save images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --class 0, or
--class 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic NMS')
    parser.add_argument('--augment', action='store_true', help='augmented inference')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default='runs/detect', help='save results to
project/name')
    parser.add_argument('--name', default='exp', help='save results to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok,
do not increment')
    parser.add_argument('--no-trace', action='store_true', help='don`t trace model')
    opt = parser.parse_args()
    print(opt)
    #check_requirements(exclude=('pycocotools', 'thop'))

    with torch.no_grad():
        if opt.update:  # update all models (to fix SourceChangeWarning)
            for opt.weights in ['yolov7.pt']:
                detect()
                strip_optimizer(opt.weights)
        else:
            detect()
```

Fig 5.5: Weapon detected (Knife)

5.5 Notifications and Alerts:

5.5.1 Email Alert:

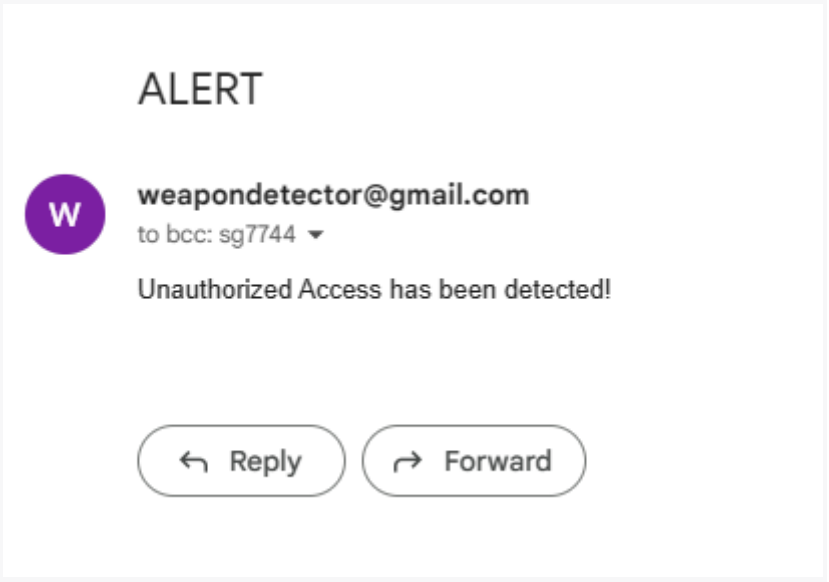


Fig 5.6: Email Alert

5.5.2 WhatsApp Alert:

```
0: Done. (593.8ms) Inference, (1.0ms) NMS
0: Done. (570.0ms) Inference, (1.0ms) NMS
0: Done. (569.9ms) Inference, (0.0ms) NMS
0: Done. (575.3ms) Inference, (1.0ms) NMS
0: Done. (573.5ms) Inference, (1.0ms) NMS
0: Done. (564.6ms) Inference, (1.5ms) NMS
In 104 Seconds WhatsApp will open and after 15 Seconds Message will be Delivered!
```

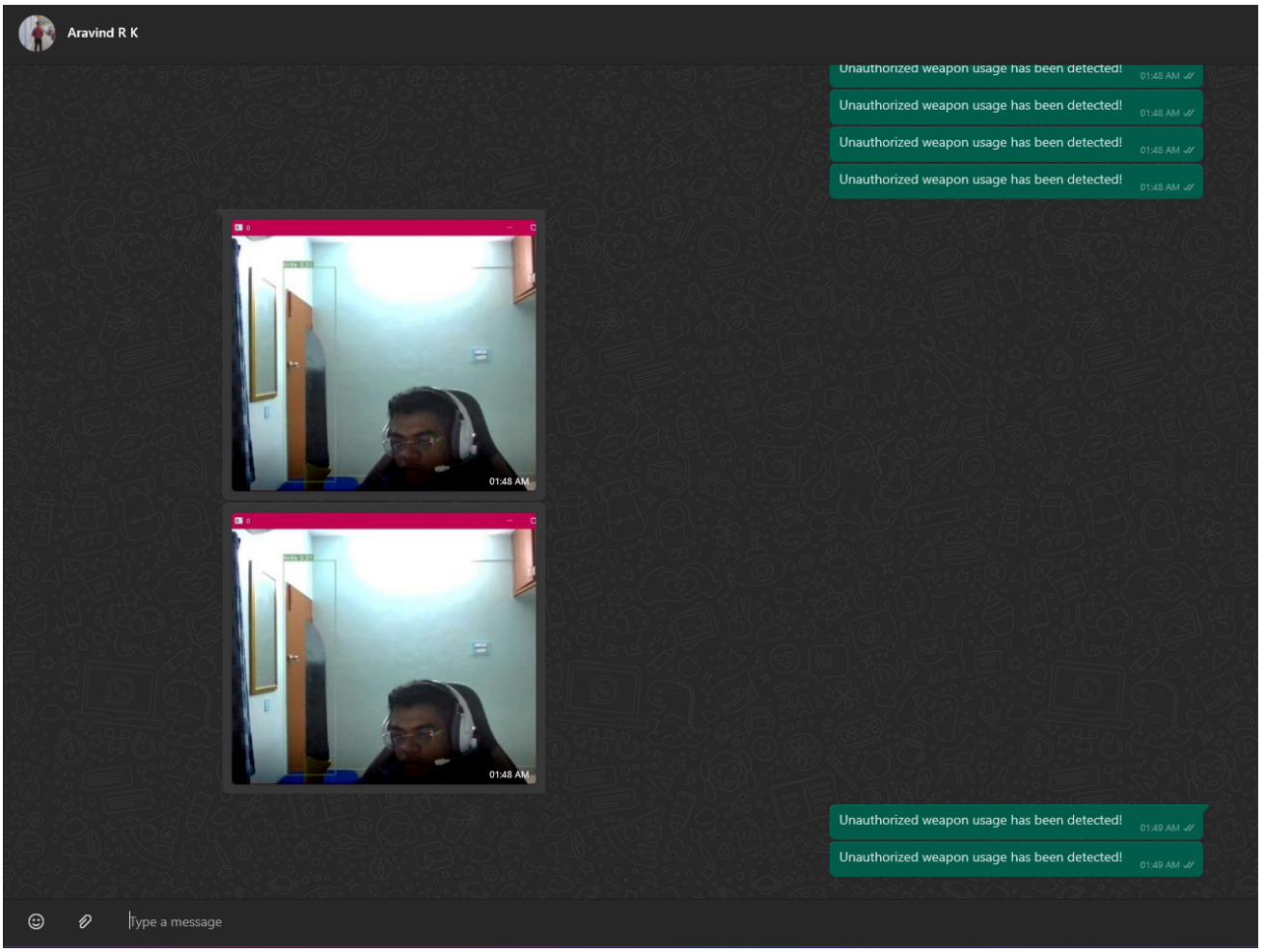


Fig 5.6: WhatsApp Alert

5.5.3 Sound Alert

Sound Alert will be triggered in the system and played simultaneously.

CHAPTER 6

RESULTS AND OBSERVATIONS

6.1 RESULTS

The YOLO v7 model for weapon detection system, which recognises the following weapon classes: Handgun, Sword, SMG, Sniper, Automatic Rifle, Bazooka, Grenade Launcher, Knife, and Shotgun, and the Haar Cascade algorithm for face recognition, have both been used to test the suggested model. The YOLO v7 model achieved an accuracy of 88.7%, average precision of 94.3%, and a recall rate of 95% for weapon identification, which is a promising outcome according to the study. Furthermore, for facial identification, the Haar Cascade method had a 96% accuracy rate. The accuracy statistic calculates the proportion of tested faces in the dataset that were properly identified. Finally, the suggested system has effectively integrated the notification and alert system, sending messages to the selected WhatsApp number and email address along with alarm sound alerts. Overall, the evaluation findings show that the suggested intelligent security system is capable of identifying faces in real-time videos and detecting weapons, and it has the potential to improve security protocols in a variety of applications, including airports, public areas, and critical infrastructure.

Sl. No	Types of Weapons	Number of Labels	Precision	Recall	Accuracy (mAP@0.5)	F1 Score
1	All	936	0.943	0.95	0.887	0.946
2	Knife	139	0.936	0.773	0.872	0.847
3	Handgun	121	0.870	0.579	0.702	0.696
4	Bazooka	65	0.705	0.625	0.651	0.663
5	Sniper	85	0.860	0.613	0.720	0.716
6	Sword	108	0.901	0.826	0.881	0.862
7	Grenade launcher	80	0.857	0.750	0.827	0.800
8	Shotgun	96	0.898	0.880	0.868	0.846
9	SMG	117	0.720	0.857	0.879	0.783

10	Automatic Rifle	125	0.506	0.700	0.684	0.588

Table 6.1.1. Evaluation Score for 9 Classes of Weapons

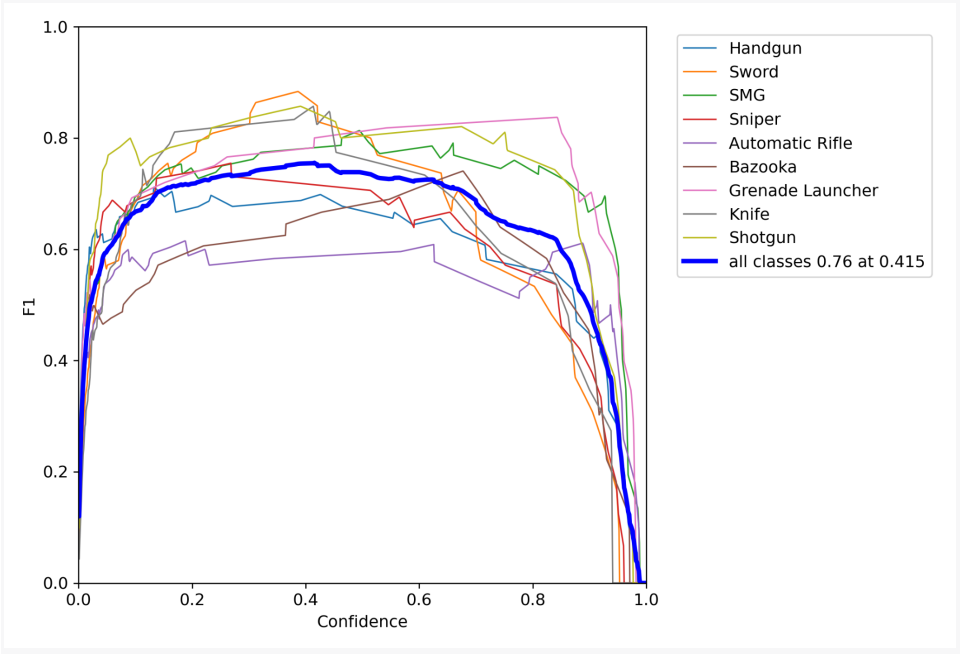
Sl.No	Epochs	Accuracy(mAP@0.5)	Precision	Recall
1	50	0.207	0.313	0.298
2	100	0.543	0.611	0.537
3	150	0.683	0.766	0.612
4	200	0.734	0.812	0.646
5	250	0.727	0.792	0.682
6	300	0.757	0.776	0.720
7	350	0.763	0.856	0.667
8	399	0.759	0.769	0.723

Table 6.1.2. Evaluation Score per Epoch

Sl.No	Weapon revealed on camera(in %)	Time Taken to Identify Weapon (in ms)
1	33%	624.1 ms
2	66%	617.6 ms
3	100%	608.4 ms

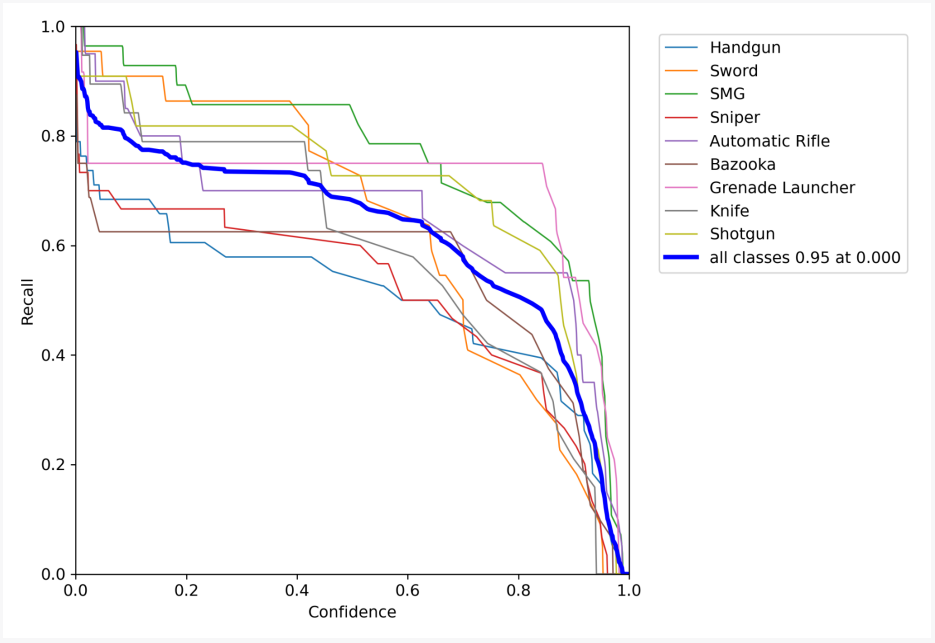
Table 6.1.3. Time taken to Identify a Weapon

6.2 GRAPHS:



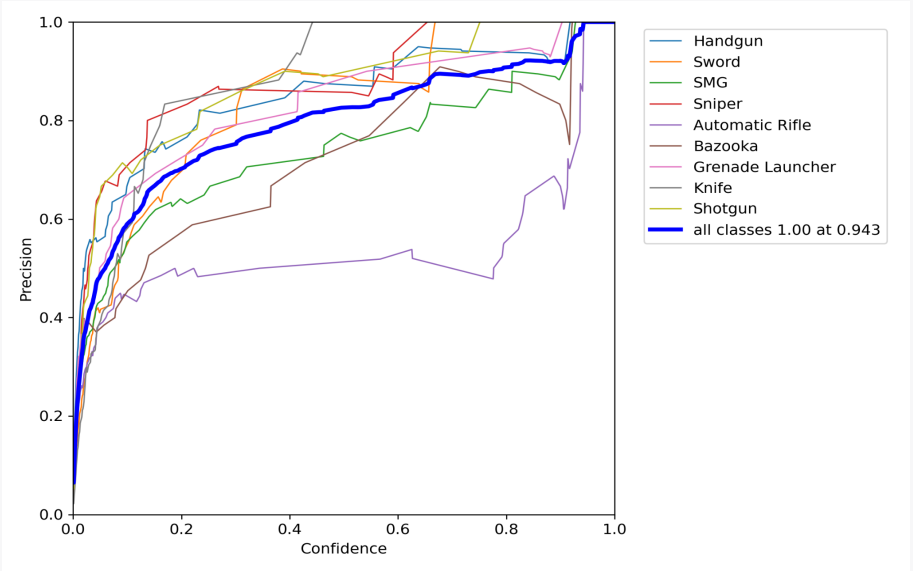
6.2.1. F1 Curve

The F1 score is a measure that combines both precision and recall, and it reaches its highest value when both precision and recall are high. When the confidence score threshold is set to 1.0, the precision of the model is high but the recall is low, which results in a high precision but a low recall. As the confidence score threshold is lowered, the recall increases but the precision decreases. This leads to an increase in the F1 score until a certain threshold, beyond which the precision starts to decrease more rapidly than the recall, resulting in a decrease in the F1 score.



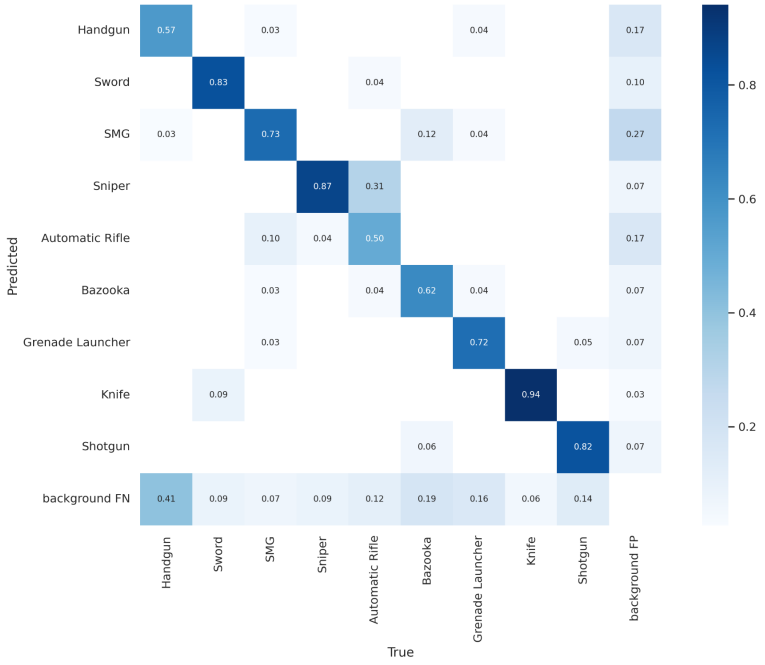
6.2.2. Recall Curve

When the confidence score threshold is set to 1.0, it means that the model is only making a prediction when it is very confident about the presence of a weapon in the image. As a result, the precision of the model increases, because it is making fewer false positive predictions at this high threshold. However, because the threshold is so high, the model may also be missing some true positive predictions, which leads to a decrease in recall. This explains the behavior of the precision and recall curves when the confidence score threshold reaches 1.0.



6.2.3. Precision Curve

The precision curve shows how precise the model's predictions are for each class as the confidence threshold is increased. The x-axis represents the confidence threshold, while the y-axis represents the precision for each class. The precision is calculated as the ratio of true positives to the total number of positive predictions (true positives plus false positives) for each class



6.2.4. Confusion Matrix

These tables show the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) for each class of weapon in your dataset. The rows represent the ground truth labels, while the columns represent the predicted labels. The diagonal elements represent the number of correctly classified instances, while the off-diagonal elements represent the misclassified instances. The confusion matrix can be used to calculate various metrics such as precision, recall, and F1 score for each class of weapon.

6.3 OBSERVATION

The combination of weapon detection and facial recognition technologies has the potential to enhance public safety and prevent potential threats in public spaces. However, the use of facial recognition raises various ethical and legal concerns such as privacy and accuracy issues. Therefore, appropriate measures need to be taken to ensure the privacy and security of individuals' biometric data and to improve the accuracy and reliability of facial recognition algorithms. In addition to ethical and legal concerns, there are also technical challenges that need to be addressed when combining weapon detection and facial recognition technologies. One of the primary challenges is the integration of different algorithms and techniques to achieve real-time and accurate detection and identification of potential threats.Moreover, the accuracy of the proposed system may be affected by various factors such as lighting, camera angles, and occlusions. Further research is needed to develop more robust algorithms that can overcome these challenges and improve the accuracy and efficiency of the system. Overall, the proposed system for real-time weapon detection using deep learning-based algorithms represents a significant advancement in public safety and security. With ongoing advances in machine learning and computer vision, we can expect to see more sophisticated and accurate weapon detection and facial recognition systems in the future.

CHAPTER 7

CONCLUSION

The proposed intelligent security system that combines YOLO v7 and Haar Cascade for weapon detection and face recognition, respectively, offers a reliable and effective solution to ensure safety and security in public places. The system is designed to detect weapons and recognize faces in real-time, and send WhatsApp and email notifications to designated users when a weapon is detected, along with an alarm alert.

The YOLO v7 algorithm was trained on a custom dataset consisting of 714 images of various weapons and achieved high accuracy and recall rate in detecting weapons. The algorithm uses a deep neural network to detect objects in images and videos. The algorithm can detect multiple objects in an image or video and can identify the type of object with high accuracy. The Haar Cascade algorithm was also able to achieve high accuracy in face recognition. The algorithm uses a set of features to detect objects in images and videos. The algorithm can detect multiple objects in an image or video and can identify the type of object with high accuracy.

The notification and alert system is an added feature that ensures that designated users are notified in real-time when a weapon is detected. This approach helps to ensure that the authorities can respond quickly to potential threats and take appropriate action to prevent any harm to the public. The system can be integrated with other security systems, such as CCTV cameras and metal detectors, to provide additional security measures.

This system has the ability to adapt to various settings and can be implemented in locations such as airports, train stations, schools, shopping malls, and public gatherings. Its purpose is to enhance security measures and provide an extra layer of safety. The proposed system has the potential to enhance public safety and prevent potential threats in public spaces. The combination of weapon detection and facial recognition technologies has the potential to enhance public safety and prevent potential threats in public spaces.

8. FUTURE WORK

Future work for this project includes several areas of improvement that can enhance the effectiveness of the proposed intelligent security system. One of the areas of improvement is to improve the accuracy of the face recognition algorithm. While the Haar Cascade algorithm used in this project achieved high accuracy, more advanced methods such as deep learning techniques can be employed to further improve the accuracy of the face recognition algorithm. Deep learning techniques can help to identify more complex facial features and improve the recognition of faces in different lighting conditions and angles.

Another area of improvement for the proposed system is to increase the detection rate of the system for detecting concealed weapons. While the YOLO v7 algorithm used in this project achieved high accuracy in detecting weapons, there is still room for improvement in detecting concealed weapons. Advanced techniques such as thermal imaging and X-ray scanning can be employed to detect concealed weapons and improve the overall effectiveness of the system.

In addition, the proposed system can be integrated with a centralized monitoring system to provide a more comprehensive security solution. A centralized monitoring system can help to monitor multiple locations simultaneously and provide real-time alerts and notifications to the authorities. This approach can help to improve the response time of the authorities and enhance the overall effectiveness of the security system.

9. REFERENCES

[1] Li, L., Li, C., & Chen, J. (2019). Research on video-based weapon detection algorithms. Journal of Physics: Conference Series, 1230(1), 012057.

[2] Zhang, Y., Huang, W., & Wu, Q. (2020). A survey on object detection in surveillance videos. IEEE Access, 8, (pp. 35459-35475).

[3] Chen, Z., & Lv, W. (2020). A survey on deep learning for object detection. Neurocomputing, 399, (pp. 23-45).

[4] Sermanet, P., Kavukcuoglu, K., Chintala, S., & LeCun, Y. (2014). Pedestrian detection with unsupervised multi-stage feature learning. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3626-3633).

[5] Hu, P., Zhang, S., & Li, B. (2019). Gun detection in surveillance videos using deep learning. International Journal of Machine Learning and Cybernetics, 10(6), (pp. 1405-1417).

[6] Cao, Y., Zhang, J., & Wang, Y. (2021). Weapon detection in surveillance videos based on deep learning. Neural Computing and Applications, 33(1), (pp. 69-77).

[7] Wang, T., Lu, K., Chen, K., & Chen, L. (2019). A survey of object detection in video. Journal of Visual Communication and Image Representation, 65, 102644.

[8] Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[9] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In the European conference on computer vision (pp. 21-37). Springer, Cham.

[10] Natarajan, P., & Nevatia, R. (2013). Detection and tracking of weapon motion paths in videos. In 2013 IEEE Conference on Computer Vision and Pattern Recognition (pp. 451-458).

[11] Zhang, C., Wang, H., & Chen, X. (2021). A survey on deep learning for object tracking in video surveillance. Neurocomputing, 443, (pp. 321-337).

[12] Liu, S., Qi, L., Wang, Z., Li, J., & Xu, W. (2019). Object detection in video with spatiotemporal sampling networks. IEEE Transactions on Circuits and Systems for Video Technology, 30(4), (pp. 1032-1046).

[13] Yin, X., Chen, W., Mei, T., & Luo, J. (2020). A comprehensive survey on object detection in videos. IEEE Transactions on Image Processing, 29,(pp. 6962-6983).

[14] Wu, C., & Nevatia, R. (2019). Detection of violent events in surveillance videos. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(12), (pp. 3039-3052).

[15] Liu, Z., Li, C., & Liu, Y. (2019). A survey of video-based human activity recognition. Neurocomputing, 338, (pp. 124-133).

APPENDIX A

CONFERENCE PRESENTATION

Our paper on **"Detect & Alert Unauthorized Weapon Usage with Haar Cascade and YOLOv7"** was presented at **IRCICD 2023 conference** held at SRM. 200+ shortlisted teams presented their papers on various fields in the conference. Our paper got accepted as paper id : IRCID_2023_66 with a plagiarism of just 7%.

International Research Conference on IoT, Cloud and Data Science (IRCICD'23)

SRM INSTITUTE OF SCIENCE & TECHNOLOGY VADAPALANI

Certificate of Participation

This is to certify that

Mr./Ms./Dr. Shriram G

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, VADAPALANI

has participated and presented a paper titled

Detect & Alert Unauthorized Weapon Usage with Haar Cascade and YOLOv7

in the International Research Conference on IoT, Cloud and Data Science (IRCICD'23), organised by the Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, Tamil Nadu, India held on 28th & 29th April 2023.

Dr. G. Paavai Anand	Dr. P. Durgadevi	Dr. S. Prasanna Devi	Dr. C . Gomathy	Dr. C. V. Jayakumar
Organizing Secretary	Organizing Secretary	HOD- CSE and Convener	VP (Academics and Placements)	Dean (CET) - VDP

International Research Conference on IoT, Cloud and Data Science (IRCICD'23)

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
VADAPALANI

Certificate of Participation

This is to certify that

Mr./Ms./Dr. V Anand Arun

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, VADAPALANI

has participated and presented a paper titled

Detect & Alert Unauthorized Weapon Usage with Haar Cascade and YOLOv7

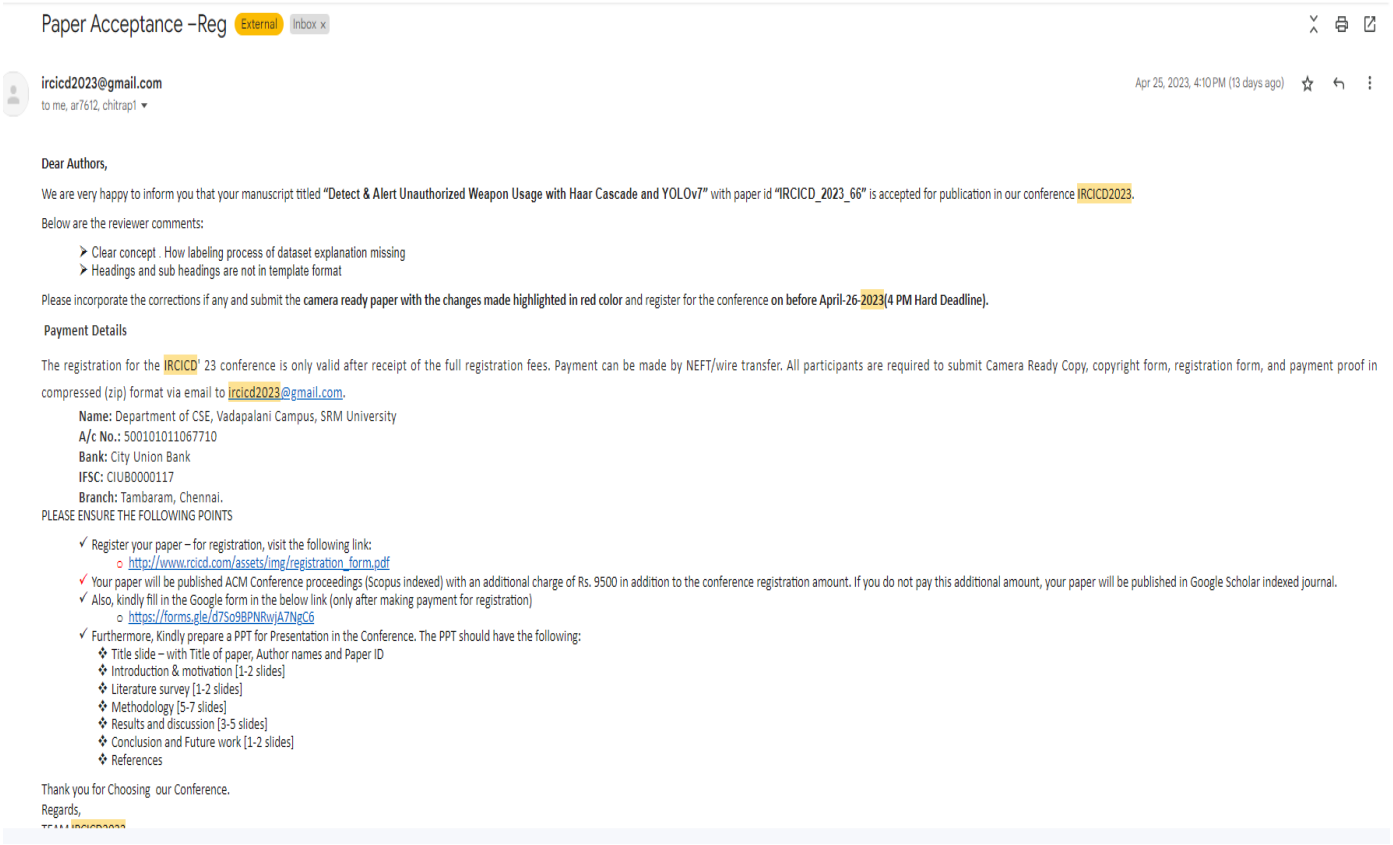
in the International Research Conference on IoT, Cloud and Data Science (IRCICD'23), organised by the Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, Tamil Nadu, India held on 28th & 29th April 2023.

Dr. G. Paavai Anand	Dr. P. Durgadevi	Dr. S. Prasanna Devi	Dr. C . Gomathy	Dr. C. V. Jayakumar
Organizing Secretary	Organizing Secretary	HOD- CSE and Convener	VP (Academics and Placements)	Dean (CET) - VDP


International Research Conference on IoT, Cloud and Data Science (IRCICD'23)
SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
VADAPALANI
Certificate of Participation
This is to certify that
Mr./Ms./Dr. Aravind R K
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, VADAPALANI
has participated and presented a paper titled
Detect & Alert Unauthorized Weapon Usage with Haar Cascade and YOLOv7
in the International Research Conference on IoT, Cloud and Data Science (IRCICD'23), organised by the Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani, Chennai, Tamil Nadu, India held on 28th & 29th April 2023.
Dr. G. Paavai Anand
Organizing Secretary
Dr. P. Durgadevi
Organizing Secretary
Dr. S. Prasanna Devi
HOD- CSE and Convener
Dr. C . Gomathy
VP (Academics and Placements)
Dr. C. V. Jayakumar
Dean (CET) - VDP

APPENDIX B

PUBLICATION DETAILS



Paper Acceptance –Reg External Inbox x

ircicd2023@gmail.com
to me, ar7612, chitrap1

Apr 25, 2023, 4:10 PM (13 days ago)

Dear Authors,

We are very happy to inform you that your manuscript titled **"Detect & Alert Unauthorized Weapon Usage with Haar Cascade and YOLOv7"** with paper id **"IRCICD_2023_66"** is accepted for publication in our conference IRCICD2023.

Below are the reviewer comments:

- Clear concept . How labeling process of dataset explanation missing
- Headings and sub headings are not in template format

Please incorporate the corrections if any and submit the **camera ready paper with the changes made highlighted in red color** and register for the conference **on before April-26-2023(4 PM Hard Deadline).**

Payment Details

The registration for the IRCICD' 23 conference is only valid after receipt of the full registration fees. Payment can be made by NEFT/wire transfer. All participants are required to submit Camera Ready Copy, copyright form, registration form, and payment proof in compressed (zip) format via email to ircicd2023@gmail.com.

Name: Department of CSE, Vadapalani Campus, SRM University
A/c No.: 500101011067710
Bank: City Union Bank
IFSC: CIUB0000117
Branch: Tambaram, Chennai.

PLEASE ENSURE THE FOLLOWING POINTS

- ✓ Register your paper – for registration, visit the following link:
  - http://www.rcicd.com/assets/img/registration_form.pdf
- ✓ Your paper will be published ACM Conference proceedings (Scopus indexed) with an additional charge of Rs. 9500 in addition to the conference registration amount. If you do not pay this additional amount, your paper will be published in Google Scholar indexed journal.
- ✓ Also, kindly fill in the Google form in the below link (only after making payment for registration)
  - https://forms.gle/d7So9BPNRwjA7NgC6
- ✓ Furthermore, Kindly prepare a PPT for Presentation in the Conference. The PPT should have the following:
  - ❖ Title slide – with Title of paper, Author names and Paper ID
  - ❖ Introduction & motivation [1-2 slides]
  - ❖ Literature survey [1-2 slides]
  - ❖ Methodology [5-7 slides]
  - ❖ Results and discussion [3-5 slides]
  - ❖ Conclusion and Future work [1-2 slides]
  - ❖ References

Thank you for Choosing our Conference.
Regards,
TEAM IRCICD2023

To Department of CSE, Vadapalani ...

₹9,500

Pay again | Split with friends

Completed

Apr 26, 2023 11:50 PM

State Bank of India
8194

UPI transaction ID
311680025247

To
•••• 7710

From: V.Anand Arun (State Bank of India)
anandarun238@oksbi

Google transaction ID
CICAgJDf5OPlJw

Powered by UPI

G Pay

APPENDIX C

PLAGIARISM REPORT

for plag check.docx

ORIGINALITY REPORT

7%	4%	4%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Lecture Notes in Computer Science, 2015. Publication	<1%
2	Submitted to CSU, San Jose State University Student Paper	<1%
3	Submitted to Nanyang Technological University Student Paper	<1%
4	Lecture Notes in Computer Science, 2014. Publication	<1%
5	www.researchgate.net Internet Source	<1%
6	worldfoodscience.com Internet Source	<1%
7	www.mdpi.com Internet Source	<1%
8	"Computer Vision – ECCV 2018", Springer Science and Business Media LLC, 2018 Publication	<1%