DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

YESHWANTRAO CHAVAN COLLEGE OF ENGINEERING

NAGPUR, MAHARASHTRA

Project Report

on

Weapon Detection Using YOLOV3(You only look once) for Smart Surveillance System

under guidance of

Prof .Shilpa Meshram

Branch: CSE (AIML) , 3rd Year (6th Sem)

Course Name: Digital Image Processing

Team Members:

a. Anshul Marathe (Roll No. 38)
b. Harsh Thakare (Roll No. 45)
c. Hrishikesh Kakde (Roll No. 47)
d. Om Mehkar (Roll No. 57)

Introduction:

Every year, gun violence takes many lives and causes enormous social, health, psychological, and financial costs. The psychological trauma that children who are exposed to violence, whether directly or indirectly, endure is very upsetting. Studies repeatedly show that handheld weapons are the main instrument used in a variety of crimes, which contributes to a broad range of societal problems. Such crimes may be prevented by early identification of disruptive conduct and close observation; nevertheless, conventional surveillance techniques frequently fall short and there is a need for more advanced measures.

The global prevalence of gun-related violence underscores the urgency for effective intervention strategies. In Pakistan alone, mass shootings result in a significant annual death toll, illustrating the dire need for improved surveillance and security measures. While human observation remains invaluable, advancements in technology, including machine learning and advanced image processing algorithms, offer promising avenues for enhancing surveillance capabilities.

Recent developments in machine learning and image processing have revolutionized surveillance systems, leveraging smart devices and networked cameras for enhanced monitoring. However, real-time detection and tracking of weapons still pose computational challenges, prompting ongoing research into robust control methods.

This project aims to address the shortcomings of traditional surveillance systems by leveraging cutting-edge technology for weapon detection. By employing computer vision techniques and deep learning, specifically YOLOv3, the system can accurately identify firearms in images in real-time. Furthermore, the project implements socket programming to demonstrate seamless integration of surveillance components, facilitating swift response to potential threats.

.

As technology continues to evolve, automated systems are poised to replace human-assisted applications, with object detection serving as a cornerstone for future advancements. Thus, the proposed system not only enhances current surveillance capabilities but also lays the groundwork for integration into future security solutions, including surveillance robots.

Objective:

To develop and implement a smart surveillance system utilizing YOLO v3 for real-time detection and accurate identification of weapons in surveillance footage, aiming to enhance public safety and security measures by proactively detecting and responding to potential threats.
This objective encompasses the utilization of YOLO v3's capabilities for multiscale detection and efficient resource utilization to enable timely and accurate identification of weapons. Additionally, the objective emphasizes the integration of the surveillance system into existing infrastructure, ensuring seamless deployment and user-friendly operation. Ultimately, the goal is to leverage advanced technology to mitigate the risks posed by firearms and other dangerous objects, contributing to overall public safety and security.

Methodology:

1. System Setup:

- **Python Installation:** The initial step involves installing Python on the system. Python is a versatile programming language widely used in machine-learning applications.
- **Library Imports:** Several essential libraries are imported for this project:
    - OpenCV (cv2): OpenCV is a popular open-source library for computer vision tasks, including image processing and video analysis. It will be used for image manipulation and object detection.
    - NumPy: NumPy is a fundamental library for scientific computing in Python. It provides efficient data structures like arrays for numerical computations used in machine learning algorithms.
    - Pandas (Optional): Pandas is a library for data analysis and manipulation. While not strictly necessary for object detection, it can be helpful for managing and analyzing training data if it's stored in a tabular format.

2. Development Environment:

- **Jupyter Notebook Installation (Optional):** While Python scripts can be used for development, Jupyter Notebook offers an interactive environment for code execution, visualization, and documentation. This can be particularly beneficial for training the model, as it allows for easier experimentation and monitoring of the training process.

3. Data Acquisition:

- **Weapon Detection Dataset:** A dataset containing labeled images of weapons is required. This dataset will be used to train the YOLOv3 model to identify weapons in future images or videos. The specific dataset used will depend on the project's needs and the types of weapons you want to detect.

4. Code Development:

- **Weapon Detection Script:** Python code is written to implement the weapon detection functionality. This script will likely leverage OpenCV and potentially other libraries to perform the following tasks:
    - Load the trained YOLOv3 model.
    - Pre-process input images or videos (e.g., resizing, normalization).
    - Perform weapon detection using the YOLOv3 model.
    - Draw bounding boxes and labels around detected weapons on the output image or video frame.
    - Display the results.

5. Model Training:

- **Jupyter Notebook (Optional):** If transfer learning is not employed (explained in step 8), a Jupyter Notebook can be used to develop and execute the training script. This script would handle:
    - Loading the weapon detection dataset.
    - Pre-processing training images (e.g., data augmentation).
    - Training the YOLOv3 model on the prepared dataset.
    - Saving the trained model weights.

6. Dataset Integration:

- **Training Script:** If model training is performed (step 5), the weapon detection dataset is integrated into the training script. The script loads the dataset images and their corresponding labels, preparing them for training the YOLOv3 model.

7. Pre-trained Model Selection:

- **YOLOv3 Model:** You Only Look Once version 3 (YOLOv3) is a real-time object detection model known for its speed and accuracy. It can be used to detect a wide range of objects, including weapons. This project leverages a pre-trained YOLOv3 model, typically provided as a weight file containing the learned parameters of the model. Pre-trained models are beneficial because they eliminate the need for extensive training from scratch, especially when dealing with complex tasks like object detection.

YOLOv3 Details:

YOLOv3 is a convolutional neural network (CNN) architecture specifically designed for object detection. It utilizes a single neural network to predict bounding boxes and class probabilities for objects in an image. This approach makes YOLOv3 faster than some alternative object detection models while still maintaining good accuracy. The pre-trained YOLOv3 model used in this project is likely trained on a large dataset of general objects and then fine-tuned on a weapon detection dataset to improve its ability to identify weapons specifically.

8. Project Organization:

- **Project Folder:** All project files, including the Python script, trained model weight file (if applicable), and configuration file (explained in step 9), are organized within a designated folder for better management.

9. Configuration File (CFG):

- **Model and Training Parameters:** A configuration file (often in the ".cfg" format) is used to define various parameters related to the YOLOv3 model and, if applicable, the training process. This file might specify details like:
  - Network architecture of the YOLOv3 model.
  - Hyperparameters for training (learning rate, batch size, etc.) if training is performed (step 5).
  - Class labels for the objects the model can detect (including weapon classes).

10. Model Readiness:

- **Trained Model:** Once the Python script, trained model weight file (if applicable), and configuration file are prepared, the weapon detection model.

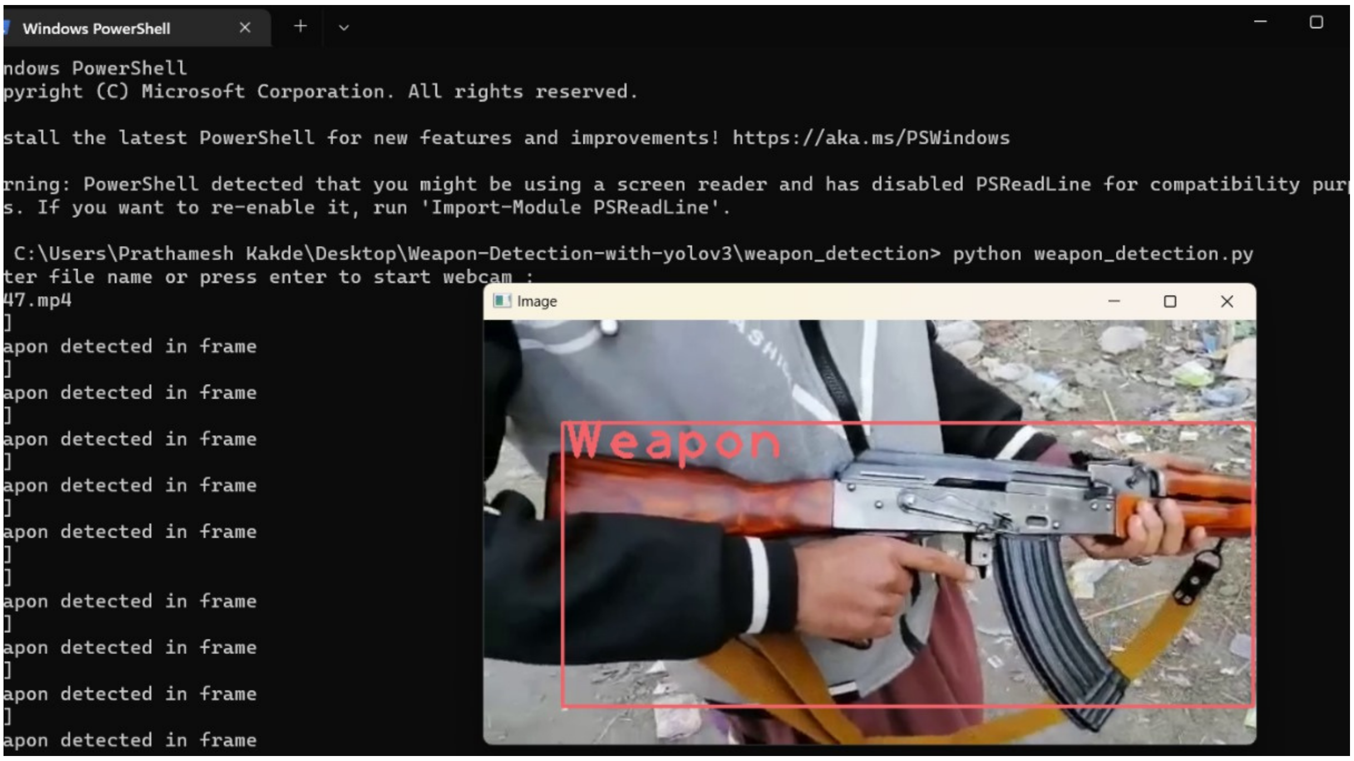
11. **Model Execution:**

- **Running the Script:** With the model components prepared (script, weights, and configuration file), the weapon detection functionality is activated by executing the Python script in the system's terminal. This initiates the program and loads the necessary elements.

12. **Input and Output:**

- **Input Flexibility:** The model can handle various input sources for weapon detection.
  - **Camera Stream:** The script can be configured to access a live camera feed, enabling real-time weapon detection in the video stream.
  - **Image File:** The script can be modified to accept a single image file as input. The model will analyze the image and identify any weapons present.
  - **Video File:** Similar to images, the script can be adapted to process a video file. Each frame of the video will be analyzed for weapons, providing weapon detection throughout the video.
- **Weapon Detection and Display:** Once the input is processed (image frames or camera feed), the YOLOv3 model performs object detection. If weapons are identified within the input, the script will:
  - Draw bounding boxes around the detected weapons on the output image or video frame. These boxes visually indicate the location and size of the weapon in the image/frame.
  - Display labels: Depending on the model configuration, labels might be displayed alongside the bounding boxes. These labels typically correspond to the specific weapon class detected (e.g., "pistol," "rifle").
- **Output Channels:** The script can potentially provide the results through two channels:
  - **Terminal Display:** Informative messages regarding the detection process or identified weapons might be printed on the terminal window for logging or user information purposes.
  - **Visual Output:** The primary output is the processed image or video frame with bounding boxes and labels overlaid on top of the detected weapons. This visual representation allows for easy identification of weapons within the input source.

Experimental Results:

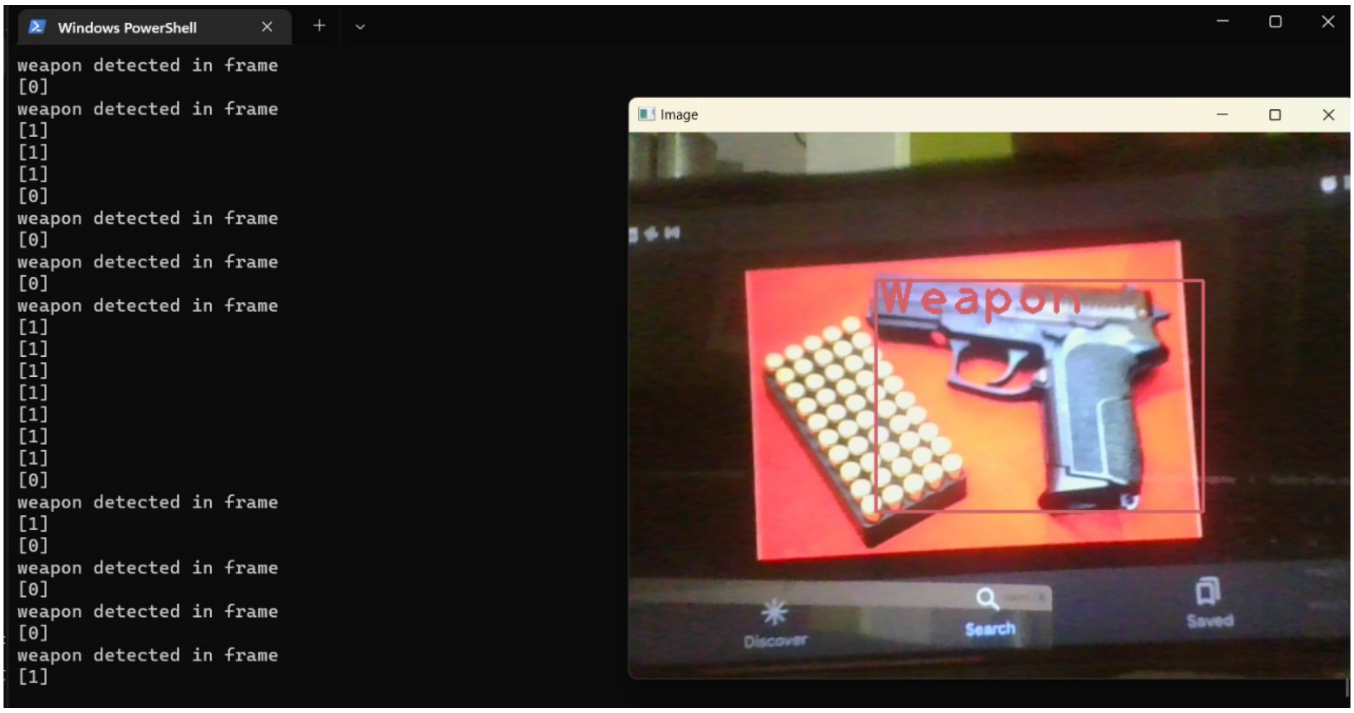


```
ndows PowerShell
pyright (C) Microsoft Corporation. All rights reserved.

stall the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

rning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility pur
s. If you want to re-enable it, run 'Import-Module PSReadLine'.

 C:\Users\Prathamesh Kakde\Desktop\Weapon-Detection-with-yolov3\weapon_detection> python weapon_detection.py
ter file name or press enter to start webcam :
47.mp4
]
apon detected in frame
]
apon detected in frame
]
apon detected in frame
]
apon detected in frame
]
apon detected in frame
]
]
apon detected in frame
]
apon detected in frame
]
apon detected in frame
]
apon detected in frame
```



```
weapon detected in frame
[0]
weapon detected in frame
[1]
[1]
[1]
[0]
weapon detected in frame
[0]
weapon detected in frame
[0]
weapon detected in frame
[1]
[1]
[1]
[1]
[1]
[1]
[1]
[0]
weapon detected in frame
[1]
[0]
weapon detected in frame
[0]
weapon detected in frame
[0]
weapon detected in frame
[1]
```

Discussion:

In our discussion, we've delved into the objective of integrating YOLO v3 into smart surveillance systems for weapon detection. This initiative is driven by the critical need to bolster public safety and security measures across diverse environments. By harnessing YOLO v3's real-time detection capabilities, renowned for their rapid alerting upon identifying weapons in surveillance footage, the aim is to proactively mitigate potential threats, particularly those involving firearms and other hazardous objects. A crucial aspect emphasized in our conversation is the paramount importance of accuracy in weapon detection. YOLO v3's sophisticated convolutional neural network (CNN) architecture enables precise identification of weapons, significantly reducing false alarms and ensuring that security resources are allocated effectively to genuine threats. Moreover, the system's multiscale detection capabilities are highlighted as a key feature, allowing for the reliable detection of weapons of various sizes and shapes, even in scenarios where they may be concealed or obscured.

Furthermore, we've underscored the efficiency in resource utilization facilitated by YOLO v3, which enables seamless integration into existing surveillance infrastructure without necessitating significant hardware upgrades. This streamlined deployment process not only enhances operational efficiency but also minimizes costs, rendering the surveillance system scalable and cost-effective. However, it's essential to acknowledge and address the challenges and limitations associated with deploying YOLO v3 for weapon detection. Factors such as sensitivity to environmental conditions and the quality of training data play a significant role in system performance. Thus, thorough testing, validation, and continuous refinement are imperative to ensure the system's reliability and effectiveness in real-world scenarios.

Advantages:

- **Real-time Detection:** YOLO v3 offers real-time weapon detection, enabling immediate threat response.
- **High Accuracy:** Its advanced CNN architecture ensures precise identification of weapons, minimizing false positives.
- **Multiscale Detection:** YOLO v3 effectively detects weapons of various sizes, enhancing coverage.

Limitations:

- **Limited Small Object Detection:** YOLO v3 may struggle with detecting small weapons.
- **Sensitivity to Environment:** Performance can be affected by lighting and clutter in surveillance footage.
- **Training Data Dependence:** Quality training data is crucial for optimal performance.

Code :

```
import cv2
import numpy as np
# Load Yolo
# Download weight file(yolov3_training_2000.weights) from this link :-
https://drive.google.com/file/d/10uJEsUpQI3EmD98iwrwzbD4e19Ps-LHZ/view?usp=sharing
net = cv2.dnn.readNet("yolov3_training_2000.weights", "yolov3_testing.cfg")
classes = ["Weapon"]
# with open("coco.names", "r") as f:
#     classes = [line.strip() for line in f.readlines()]
# layer_names = net.getLayerNames()
# output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
output_layer_names = net.getUnconnectedOutLayersNames()
colors = np.random.uniform(0, 255, size=(len(classes), 3))
```

```
# Loading image
# img = cv2.imread("room_ser.jpg")
# img = cv2.resize(img, None, fx=0.4, fy=0.4)
# Enter file name for example "ak47.mp4" or press "Enter" to start webcam
def value():
    val = input("Enter file name or press enter to start webcam : \n")
    if val == "":
        val = 0
    return val
# for video capture
cap = cv2.VideoCapture(value())
# val = cv2.VideoCapture()
while True:
    _, img = cap.read()
    if not _:
        print("Error: Failed to read a frame from the video source.")
        break
    height, width, channels = img.shape
    # width = 512
    # height = 512

    # Detecting objects
    blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)

    net.setInput(blob)
    outs = net.forward(output_layer_names)
 # Showing information on the screen
    class_ids = []
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5:
```

```
# Object detected
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            w = int(detection[2] * width)
            h = int(detection[3] * height)

            # Rectangle coordinates
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)

            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)

   indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
   print(indexes)
   if indexes == 0: print("weapon detected in frame")
   font = cv2.FONT_HERSHEY_PLAIN
   for i in range(len(boxes)):
      if i in indexes:
         x, y, w, h = boxes[i]
         label = str(classes[class_ids[i]])
         color = colors[class_ids[i]]
         cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
         cv2.putText(img, label, (x, y + 30), font, 3, color, 3)

   # frame = cv2.resize(img, (width, height), interpolation=cv2.INTER_AREA)
   cv2.imshow("Image", img)
   key = cv2.waitKey(1)
   if key == 27:
      break
cap.release()
cv2.destroyAllWindows()
```

Conclusion:

This study focuses on implementing and training the advanced YOLO V3 object detection model using a curated dataset specifically for weapon identification. Our proposed model aims to equip machines or robots with the capability to detect and alert human administrators promptly upon recognizing firearms. Experimental findings demonstrate that the trained YOLO V3 outperforms the previous YOLO V2 model and demands fewer computational resources. There's an urgent necessity to enhance existing surveillance capabilities by embracing updated technologies to augment human operators' effectiveness. As low-cost storage, improved video infrastructure, and advanced video processing technologies become more accessible, there's a trajectory towards the complete replacement of current surveillance infrastructure with smarter, more cost-effective solutions. Eventually, with the proliferation of affordable computing resources, sophisticated video infrastructure, and enhanced video processing technologies, digital monitoring systems, including robotic surveillance, are poised to supplant conventional surveillance systems entirely.

References:

1. Sanam Narejo , Bishwajeet Pandey , Doris Esenarro vargas , Ciro Rodriguez , and M. Rizwan Anjum ,”**Weapon Detection Using YOLO V3 for Smart Surveillance System**” *Hindawi Mathematical Problems in Engineering Volume 2021,*
2. S. B. Kibria and M. S. Hasan, “**An analysis of feature extraction and classification algorithms for dangerous object detection,**” in Proceedings of the 2017 2nd International Conference on Electrical & Electronic Engineering (ICEEE), pp. 1–4, IEEE, Rajshahi, Bangladesh, December 2017.

3. V. Babanne, N. S. Mahajan, R. L. Sharma, and P. P. Gargate, "**Machine learning-based smart surveillance system,**" in Proceedings of the 2019 1st International Conference on I-SMAC (IoT in Social, Mobile, Analytics, and Cloud), pp. 84–86, IEEE, Palladam, India, December 2019.

4. S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B.-Y. Choi, and T. Faughnan, "**Smart surveillance as an edge network service:** from harr-cascade, SVM to a lightweight CNN," in Proceedings of the 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), pp. 255-265, Philadelphia, PA, USA, April 2018.

5. R. Xu, S. Y. Nikouei, Y. Chen et al., "**Real-time human objects tracking for smart surveillance at the edge,"** in Proceedings of the 2018 IEEE International Conference on Communications (ICC), pp. 1–6, Kansas City, MO, USA, May 2018.

6. S. Ahmed, H. Wang, and Y. Tian, "**Adaptive high-order terminal sliding mode control based on time delay estimation for the robotic manipulators with backlash hysteresis,"** IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 2, pp. 1128–1137, 2021.

7. A. Farhadi and R. Joseph, "**Yolov3: an incremental improvement," Computer Vision and Pattern Recognition**, 2018.

8. L. Pang, H. Liu, Y. Chen, and J. Miao, **"Real-time concealed object detection from passive millimeter wave images based on the YOLOv3 algorithm**," Sensors, vol. 20, no. 6, p. 1678, 2020.

9. A. Warsi, M. Abdullah, M. N. Husen, M. Yahya, S. Khan, and N. Jawaid, "**Gun detection system using YOLOv3,**" in Proceedings of the 2019 IEEE International Conference on Smart Instrumentation, Measurement, and Application (ICSIMA), pp. 1–4, IEEE, Kuala Lumpur, Malaysia, August 2019.