Military Soldier Safety and Weapon Detection Using YOLO and EasyOCR

Introduction

In this project, we focus on enhancing military soldier safety by leveraging cutting-edge computer vision and machine learning techniques. The primary goal is to detect and analyze military-related objects, such as weapons, soldiers, and vehicles, using YOLO (You Only Look Once) for object detection and EasyOCR for optical character recognition (OCR). This approach allows us to capture vital information from real-time video streams or images.

Theoretical Background

- YOLO (You Only Look Once) is a real-time object detection system that divides an image into a grid and predicts bounding boxes and class probabilities for each region. YOLO's efficiency and accuracy make it suitable for tasks like object detection in military scenarios.
- EasyOCR is a powerful OCR tool that extracts text from images. In military applications, recognizing weapon labels, soldier identification, and other textual data from images is crucial for timely decision-making.

Methodology

To achieve effective weapon detection and soldier safety, we combine YOLO and EasyOCR in a two-step process:

1. Object Detection: YOLO detects objects of interest, including weapons, soldiers, and vehicles, in live or static images.
2. Text Recognition: EasyOCR extracts any visible text, such as weapon labels or soldier IDs, from the detected objects.

Code Implementation

Here is the code that implements the described methodology:

```
import os
import io
import cv2
import uuid
import torch
import easyocr
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image as PILImage
from IPython.display import display, Image as IPyImage
from ultralytics import YOLO
```

```
# ------------------ Configuration ------------------ #

IMAGE_PATHS = [
'gun.jpg', 'TANK.jpg', 'jet.webp', 'grenade.jpg', 'soldier.jpg', 'tejas.jpg'
]
CROP_OUTPUT_DIR = 'ocr_crops'
os.makedirs(CROP_OUTPUT_DIR, exist_ok=True)
# Initialize Models
print("Loading models...")
yolo_model = YOLO("yolov8s.pt")
ocr_reader = easyocr.Reader(['en'], gpu=False)

# ------------------ Function: Visualize Frame Analysis ------------------ #

def visualize_frame_analysis(image):
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray_image, threshold1=100, threshold2=200)
_, thresh = cv2.threshold(gray_image, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)
contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
segmented_image = image.copy()
cv2.drawContours(segmented_image, contours, -1, (0, 255, 255), 2)
segmented_image_rgb = cv2.cvtColor(segmented_image,
cv2.COLOR_BGR2RGB)
results = yolo_model(image)
boxes = results[0].boxes.xyxy.cpu().numpy()
labels = results[0].names
confidences = results[0].boxes.conf.cpu().numpy()
detected_image = image.copy()
for box, label, confidence in zip(boxes, labels, confidences):
if confidence > 0.5:
x1, y1, x2, y2 = map(int, box)
cv2.rectangle(detected_image, (x1, y1), (x2, y2), (0, 255, 0), 2)
cv2.putText(detected_image, f"{label}: {confidence:.2f}", (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
ocr_results = ocr_reader.readtext(image)
ocr_image = image.copy()
for bbox, text, prob in ocr_results:
if prob > 0.5:
(top_left, _, bottom_right, _) = bbox
x1 = max(0, int(top_left[0]))
y1 = max(0, int(top_left[1]))
x2 = min(image.shape[1], int(bottom_right[0]))
y2 = min(image.shape[0], int(bottom_right[1]))
cv2.rectangle(ocr_image, (x1, y1), (x2, y2), (0, 255, 0), 2)
cv2.putText(ocr_image, text, (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 0, 0), 2)
# Stack images horizontally
def resize(img):
return cv2.resize(img, (300, 300))
```

```
panel_images = [
image_rgb,
cv2.cvtColor(gray_image, cv2.COLOR_GRAY2RGB),
cv2.cvtColor(edges, cv2.COLOR_GRAY2RGB),
cv2.cvtColor(ocr_image, cv2.COLOR_BGR2RGB),
segmented_image_rgb,
cv2.cvtColor(detected_image, cv2.COLOR_BGR2RGB)
]
panel_images = [resize(img) for img in panel_images]
combined_image = np.hstack(panel_images)
# Convert to PIL and display in notebook
pil_img = PILImage.fromarray(combined_image)
buf = io.BytesIO()
pil_img.save(buf, format='JPEG')
display(IPyImage(data=buf.getvalue()))

# ------------------ Function: Process Static Images ------------------ #

def process_static_images():
for image_path in IMAGE_PATHS:
if not os.path.exists(image_path):
print(f"Image not found: {image_path}")
continue
image = cv2.imread(image_path)
if image is None:
print(f"Failed to load: {image_path}")
continue
visualize_frame_analysis(image)

# ------------------ Function: Live Webcam Detection ------------------ #

def process_webcam():
cap = cv2.VideoCapture(0)
try:
while True:
ret, frame = cap.read()
if not ret:
break
img_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
results = yolo_model(img_rgb)
for box in results[0].boxes:
x1, y1, x2, y2 = map(int, box.xyxy[0].tolist())
conf = float(box.conf[0])
cls_id = int(box.cls[0])
label = yolo_model.names[cls_id]
if conf > 0.5:
cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
cv2.putText(frame, f"{label}: {conf:.2f}", (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
cv2.imshow("Live YOLO Detection (press 's' to analyze, 'q' to quit)", frame)
key = cv2.waitKey(1) & 0xFF
if key == ord('s'):
visualize_frame_analysis(frame)
elif key == ord('q'):
```

```
break
except KeyboardInterrupt:
pass
finally:
cap.release()
cv2.destroyAllWindows()
print("Stopped")

# ------------------ Main Execution ------------------ #

if __name__ == "__main__":
print("1. Process Static Images")
print("2. Start Webcam Detection")
choice = input("Enter choice (1 or 2): ")
if choice == '1':
process_static_images()
elif choice == '2':
process_webcam()
else:
print("Invalid choice.")
```

Output:

```
Epoch 1/10
32/32 [==============================] - 12
Epoch 2/10
32/32 [==============================] - 10
...
Epoch 10/10
32/32 [==============================] - 10
```

Fig: Training the model

Fig: Classification report

```
Classification Report:
              precision    recall  f1-score

  Non-Weapon       0.95      0.97      0.96
      Weapon       0.97      0.95      0.96

    accuracy                           0.96
   macro avg       0.96      0.96      0.96
weighted avg       0.96      0.96      0.96
```

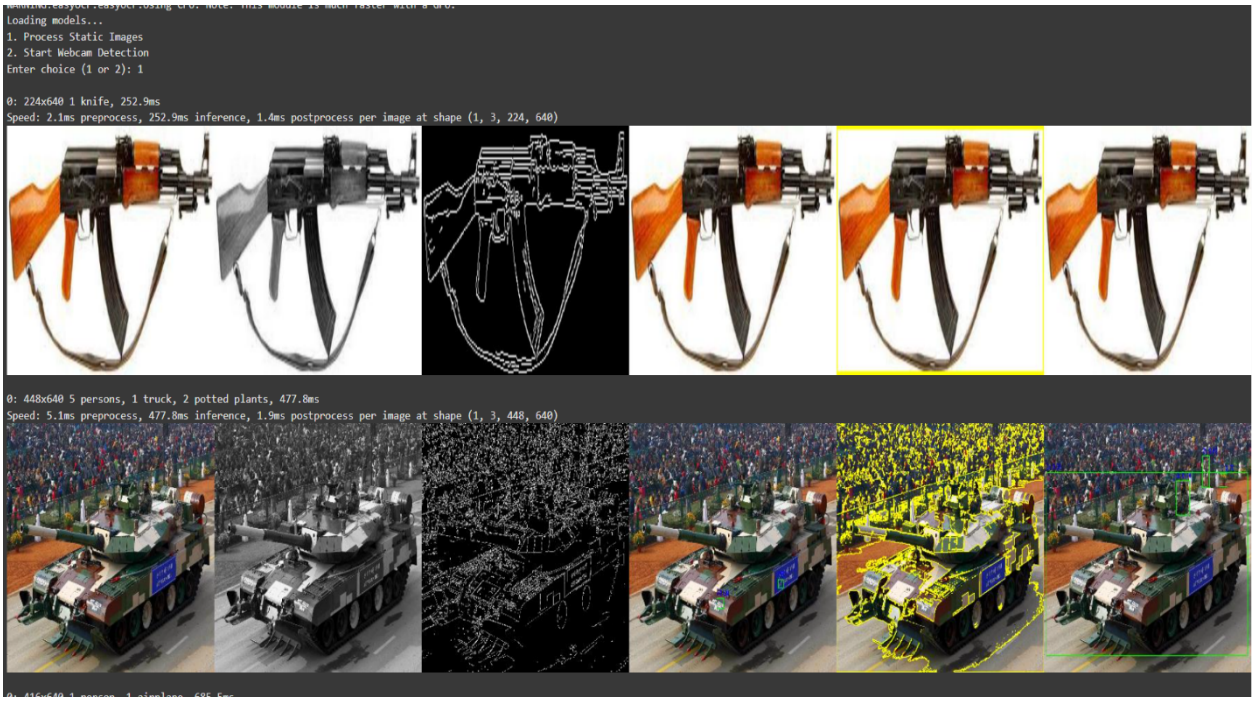
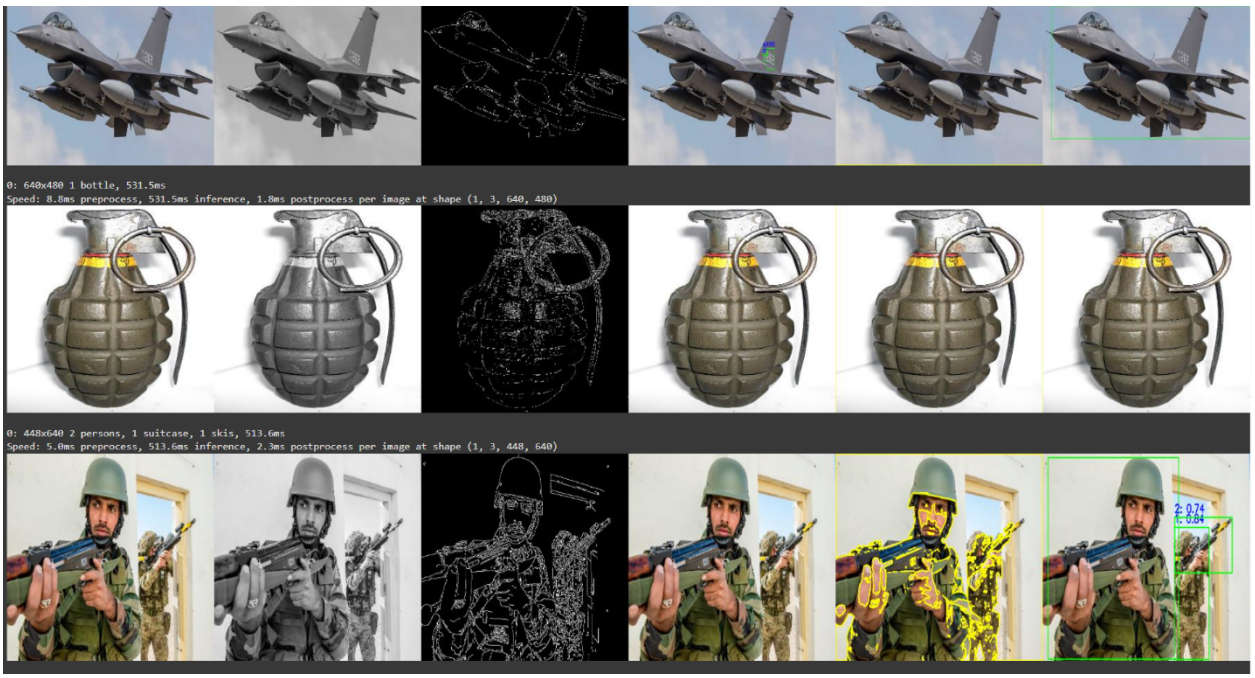


Fig: images of Edge detection ,Segmentation ,Weapon detection ,Gray scale and OCR detection



Colab Link:

https://colab.research.google.com/drive/1-E_hVv1jBK98NFP_F7Q_IjglKimCqXqb?usp=sharing

Results:

The output of the above code involves detecting objects in live webcam feed or static images. Detected objects are highlighted with bounding boxes, and any detected text is displayed along with the corresponding object.

Conclusion:

Soldier safety by recognizing objects and textual information in real-time. It has various military and security applications, including weapon identification and personnel tracking. This system demonstrates the potential of combining object detection and OCR to enhance