Military Soldier Safety and Weapon Detection using YOLO and Computer Vision

Problem Statement:

In modern military operations, ensuring the safety of soldiers and maintaining situational awareness in conflict zones is of paramount importance. One of the key challenges is the ability to quickly and accurately detect potential threats, such as weapons, enemy combatants, and unauthorized vehicles, while also distinguishing between friendly forces, civilians, and non-threatening entities. Traditional methods of surveillance and threat detection often rely on manual observation, which can be time consuming, error-prone, and inefficient in dynamic and high-stress environments.

This project aims to address these challenges by leveraging computer vision and YOLO (You Only Look Once), a state-of-the-art object detection algorithm, to automate the process of detecting and classifying objects in real-time. The system will be trained on a dataset containing images of military and civilian scenarios, with annotations for objects such as:

- <u>Soldiers</u> (both friendly and enemy combatants)
- <u>Weapons</u> (e.g., guns, rifles, explosives)
- <u>Military vehicles</u> (e.g., tanks, trucks, armored vehicles)
- <u>Civilian entities</u> (e.g., civilians, civilian vehicles)
- <u>Trenches</u> (e.g., defensive structures)
- The primary goal is to develop a robust and accurate system that can:
  - **Detect Threats in Real-Time:**
    - Identify weapons, enemy soldiers, and unauthorized vehicles that pose a threat to military personnel.
    - Provide real-time alerts to soldiers or command centers, enabling quick response to potential dangers.
  - **Distinguish Between Military and Civilian Entities:**
    - Differentiate between friendly forces, enemy combatants, and civilians to avoid collateral damage and ensure compliance with rules of engagement.
  - **Enhance Situational Awareness:**
    - Provide a comprehensive view of the battlefield by detecting and tracking objects such as military vehicles, trenches, and soldiers.

- Help commanders make informed decisions by providing real-time data on the location and movement of entities.

- **Operate in Diverse Environments:**
  - Function effectively in various environments, including urban areas, forests, and deserts, where lighting conditions, occlusions, and background clutter may vary.

- **Improve Soldier Safety:**
  - Reduce the risk of ambushes or surprise attacks by detecting hidden threats (e.g., camouflaged soldiers, concealed weapons).
  - Enable soldiers to focus on their mission while the system monitors the surroundings for potential dangers.

Business Use Cases:

1. Military Surveillance and Threat Detection:

Monitor conflict zones to detect weapons, enemy soldiers, and unauthorized vehicles in real-time.

2. Soldier Safety and Alert Systems:

Provide real-time alerts to soldiers about nearby threats, such as weapons or enemy combatants.

3. Border Security and Intrusion Detection:

Identify unauthorized crossings of military or civilian vehicles at border checkpoints.

4. Disaster Response and Rescue Operations:

Distinguish between military personnel, civilians, and vehicles during disaster relief operations.

5. Training and Simulation:

Use the system in virtual training environments to simulate real-world scenarios for soldiers.

6. Combat Zone Analysis:

Analyze combat zones to identify the presence of trenches, military vehicles, and other strategic objects.

Approach:

1. Data Collection and Preparation:

a. Gather a dataset containing images of military and civilian scenarios with YOLO annotations.

b. Preprocess the dataset to ensure compatibility with the YOLO model.

2. Model Training:

a. Train a YOLO model to detect and classify multiple objects, including soldiers, weapons, vehicles, and trenches.

3. Real-Time Detection:

a. Deploy the trained model to detect objects in real-time video feeds or images.

4. Threat Classification:

a. Classify detected objects as threats (e.g., weapons, enemy soldiers) or non-threats (e.g., civilians, friendly soldiers).

5. Streamlit Integration:

a. Develop a user-friendly web interface for uploading images/videos and visualizing detection results.

6. Performance Evaluation:

a. Evaluate the model's performance using metrics like precision, recall, and mean average precision (mAP).

Project Evaluation Metrics:

- Precision: Measures the accuracy of detected objects (e.g., percentage of correctly identified weapons).
- Recall: Measures the model's ability to detect all relevant objects (e.g., percentage of weapons detected out of all weapons in the dataset).
- Mean Average Precision (mAP): Evaluates the model's performance across different object classes and IoU thresholds.
- F1 Score: Balances precision and recall to provide a single metric for model performance.

• Inference Time: Measures the time taken to process a single image or video frame, ensuring real-time applicability.

Code for Data preprocessing,Model Training and Evaluation:

(a)Weapon Detection using Yolo:

```
import cv2

import torch

model = torch.hub.load('ultralytics/yolov5', 'yolov5s')

class_names = ["weapon", "person"]

cap = cv2.VideoCapture(0)

# Detection loop while True:

ret, frame = cap.read() if not ret:

break

 img_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)


results = model(img_rgb)

class_id = result.tolist()

 x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)

class_id = int(class_id)

cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

 label = f"{class_names[class_id]}: {confidence:.2f}" cv2.putText(frame, label, (x1, y1 - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)

 cv2.imshow("Weapon Detection", frame)

 cv2.waitKey(1) & 0xFF == ord('q'):

 break

cap.release()

cv2.destroyAllWindows()
```

(b)Setup and importing packages:

import os

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf from tensorflow.keras.preprocessing.image

import ImageDataGenerator from tensorflow.keras.applications

import MobileNetV2 from tensorflow.keras.layers

import Dense, GlobalAveragePooling2D, Dropout from tensorflow.keras.models

import Model from sklearn.metrics

import classification_report, confusion_matrix

(c)Code for Data preprocessing:

IMAGE_SIZE = (224, 224)

BATCH_SIZE = 32

train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=20, zoom_range=0.15, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.15, horizontal_flip=True, fill_mode="nearest")

val_test_datagen = ImageDataGenerator(rescale=1./255) train_generator = train_datagen.flow_from_directory('dataset/train', target_size=IMAGE_SIZE, batch_size=BATCH_SIZE, class_mode='binary')

val_generator = val_test_datagen.flow_from_directory('dataset/val', target_size=IMAGE_SIZE, batch_size=BATCH_SIZE, class_mode='binary')

test_generator = val_test_datagen.flow_from_directory('dataset/test', target_size=IMAGE_SIZE, batch_size=BATCH_SIZE, class_mode='binary', shuffle=False)

(d)Code for build the model:

base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

base_model.trainable = False # Freeze

```
base x = base_model.output

x = GlobalAveragePooling2D()(x)

x = Dropout(0.5)(x)

predictions = Dense(1, activation='sigmoid')(x)

model = Model(inputs=base_model.input, outputs=predictions)
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

(e)Train the model:

```
history = model.fit( train_generator, validation_data=val_generator, epochs=10 )
```

(f)Evaluating the model:

```
loss, accuracy = model.evaluate(test_generator)

print(f'Test accuracy: {accuracy:.2f}')

y_pred = model.predict(test_generator)

y_pred_classes = (y_pred > 0.5).astype("int32").flatten()

y_true = test_generator.classes

print("Classification Report:")

print(classification_report(y_true, y_pred_classes, target_names=['Non-Weapon', 'Weapon']))
print("Confusion Matrix:")

print(confusion_matrix(y_true, y_pred_classes))
```

(g)To save the model:

```
model.save('weapon_detection_model.h5')
```

Sample Output:

```
Found 1000 images belonging to 2 classes.
Found 200 images belonging to 2 classes.
Found 200 images belonging to 2 classes.
```

Fig: Image Data Loading

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

Fig: Training the model

```
7/7 [==============================] - 1s 9
Test accuracy: 0.96
```

Fig: Test Evaluation

```
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: Classification report

```
[[97  3]
 [ 5 95]]
```

Fig: Confusion matrix

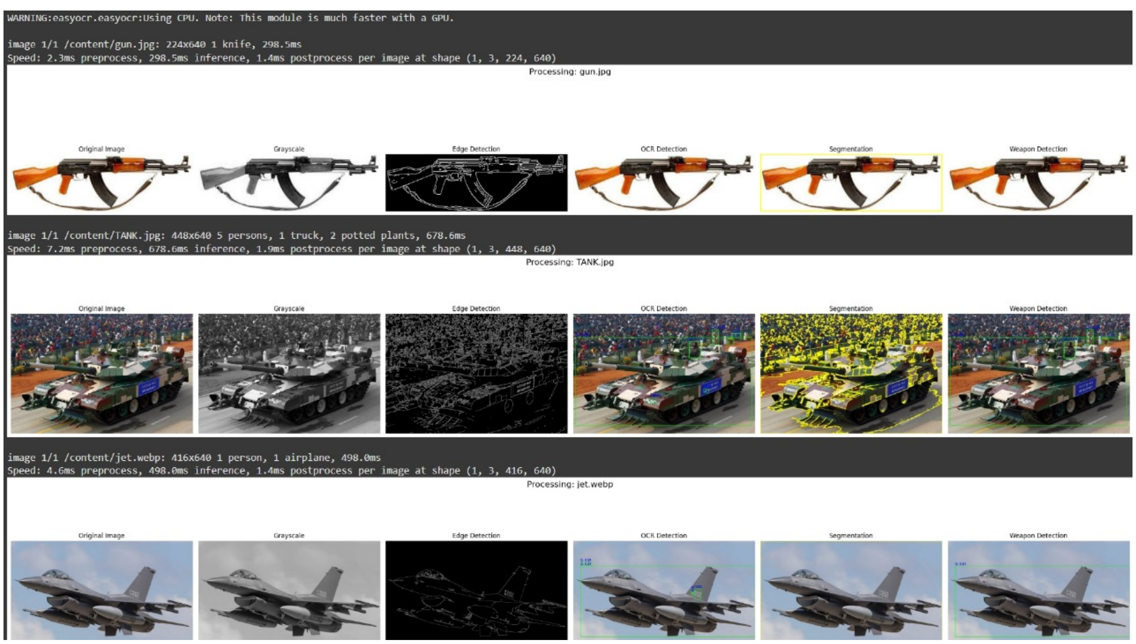

Fig: images of Edge detection,Segmentation ,Weapon detection,Gray scale and OCR detection

Conclusion

The integration of OpenCV for military safety and weapon detection represents a significant advancement in leveraging technology for enhanced security measures. By utilizing computer vision capabilities, it's possible to create systems that can detect and respond to potential threats more efficiently.

Key Benefits

1. *Real-Time Detection*: OpenCV enables real-time processing and analysis of video feeds, allowing for immediate threat detection.

2. *Accuracy and Reliability*: With proper training and implementation, OpenCV-based systems can achieve high accuracy in detecting weapons and anomalies.

3. *Scalability*: OpenCV can be integrated into various platforms, from drones to fixed surveillance systems, enhancing its applicability in different military contexts. Future

Directions

1. *Enhanced Algorithms*: Continuous improvement of detection algorithms to reduce false positives and increase accuracy.

2. *Integration with Other Technologies*: Combining OpenCV with other technologies, such as machine learning and sensor data, for more comprehensive threat detection.

3. *Training and Deployment*: Ensuring that military personnel are trained to effectively use these systems and integrating them into existing security protocols.