FYP

Emergency Alert App
Project Report

Supervisor

Almas Alyas

Group Members

Muhammad Hamza Mughal (Lead) 201400164

Muhammad Fahad Zafar

Chd Salman Ahsan

Usman Sheikh

Institution

GIFT University Gujranwala

Table of Contents

Introduction

Background

In the Today 'Society, people who need immediate help usually have telephone hotline services, which could cause delays and result in the loss of vital time in urgent situations. The location and specifics of the emergency were frequently communicated over the phone, which led to errors and inefficiencies that delayed the arrival of emergency personnel on time. Another issue was the widespread deployment of CCTV cameras in homes and workplaces, which frequently resulted in a lack of ongoing oversight of the recorded material or it is a very tough job to monitor real time footage for a person for security. So, without real-time activity monitoring, incidents can happen without prompt notice or action, exposing people and property at risk from a variety of dangers.

Project Overview

In order to overcome these obstacles (discuss in problem statement), we creating an application with a Suspicious Activity detection through CCTV live footage with Machine Learning model, giving consumers instant access to all necessary emergency services, contacts with Call and default message, along with live location and integration of Alarm (IOT Device) for trigger at the time of critical situation. This streamlined procedure seeks to improve overall efficiency, security, and speed while drastically reducing reaction times,

Activities Includes

1. Wall Climbing
2. Covered Face
3. Weapon Detection

Existing Apps

Existing Apps	Specious Activity Detection	IOT Devices	Call to Emergency Services	Give Alert To Contacts	Live Location Sending with default Message
Emergency Plus	✗	✗	✓	✗	✓
SOS Safety App	✗	✗	✓	✓	✗
Personal Safety	✗	✗	✓	✓	✓
EAS(OUR APP)	✓	✓	✓	✓	✓

System Architecture

Technologies Used

- Frontend: React Native for mobile app development.
- Backend: Firebase for cloud storage and Node.js for server-side operations.
- Machine Learning: YOLOv8 model for real-time activity detection.
- IoT: ESP32 and Bluetooth speaker for alarm triggering.
- Communication: WhatsApp API for messaging due to telecommunication restrictions.
- RTSP Protocol: Used for streaming live video from IP cameras.
- Google Maps API: For reverse geocoding to retrieve readable addresses from coordinates.

System Flow

1. Video Feed: The IP camera sends live footage to the mobile app via RTSP protocol.
2. Machine Learning Detection: The YOLOv8 model analyzes video frames to detect suspicious activities.
3. Alarm Triggering: Based on detection, an alarm is triggered through either Bluetooth speakers or an ESP32 device.
4. SOS Alerts: Users can manually send SOS alerts along with live location to their contacts.
5. Location Services: The app fetches the user's current location and performs reverse geocoding to convert lat/lon to a human-readable address.

Implementation

5.1 IP Camera Integration

One of the key challenges was fetching real-time footage from IP cameras into a React Native app. Initially, an external app was needed to connect the IP camera, but after extensive research, we managed to implement a custom player in React Native that uses the RTSP protocol to fetch and play live video streams.

5.2 Alarm Triggering

We explored multiple approaches for triggering alarms:

1. *Bluetooth Speakers:* The app connects with a Bluetooth speaker to play pre-recorded alarm sounds.
2. *ESP32 Alarm System:* We developed an IoT device using ESP32 that generates noise when triggered by a request sent from the app.
3. *Node.js Server Trigger:* The most reliable approach was to connect a speaker to a Node.js server that listens for requests sent from the app. When the alarm button is pressed, the app sends a request to the server, which in turn triggers the alarm.

5.3 Automatic Messaging and Calls

Automated messaging and calls posed another significant challenge due to restrictions from telecommunication companies. As a solution, we integrated WhatsApp messaging, allowing users to send messages with current locations to emergency contacts in just three clicks.

5.4 Location Services

The app uses GPS to fetch the user's current location. Implementing reverse geocoding to convert latitude and longitude into a human-readable address took significant time. With the help of the Google Maps API, we ensured the app generates accurate and readable addresses from coordinates.

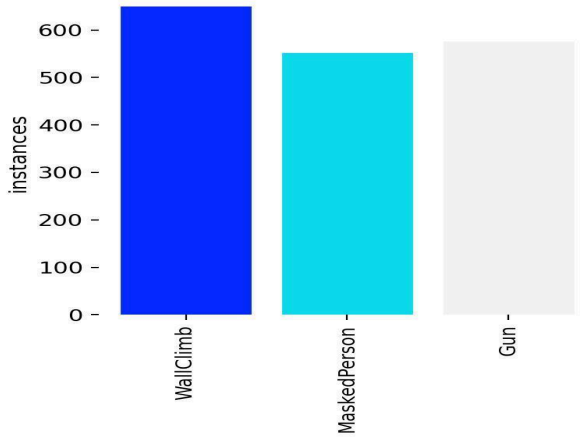
5.5 Suspicious Activity Detection (YOLOv8 Model)

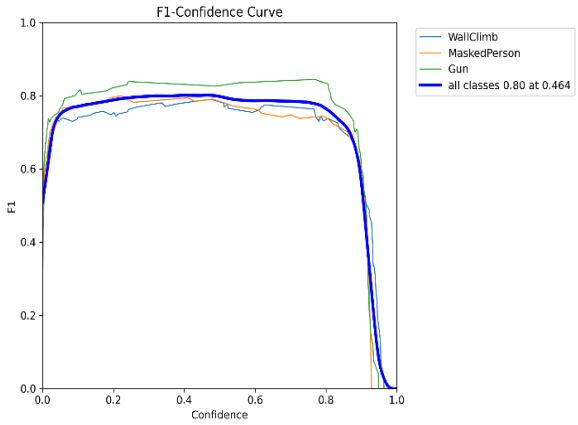
We trained the YOLOv8 model for detecting specific activities:

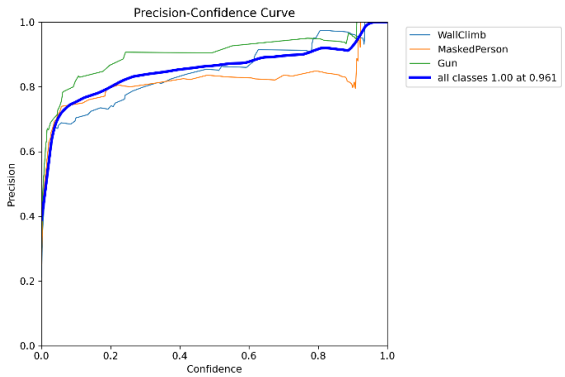
- Wall climbing
- Covered faces
- Weapon detection

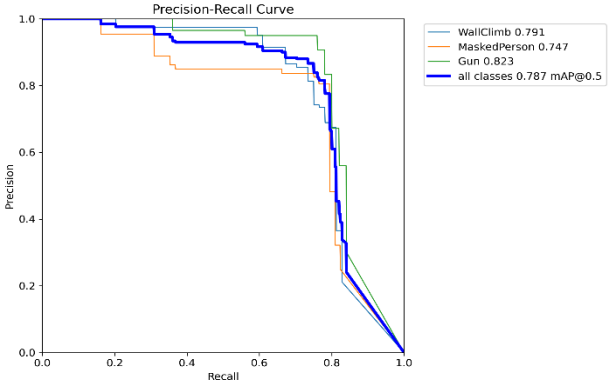
Due to a lack of relevant datasets, we collected images manually and labeled them in chunks. The model underwent several training cycles to achieve accuracy. However, the initial model sent multiple notifications (one per detected frame), overwhelming the user. We resolved this by setting intervals between notifications to avoid spamming.

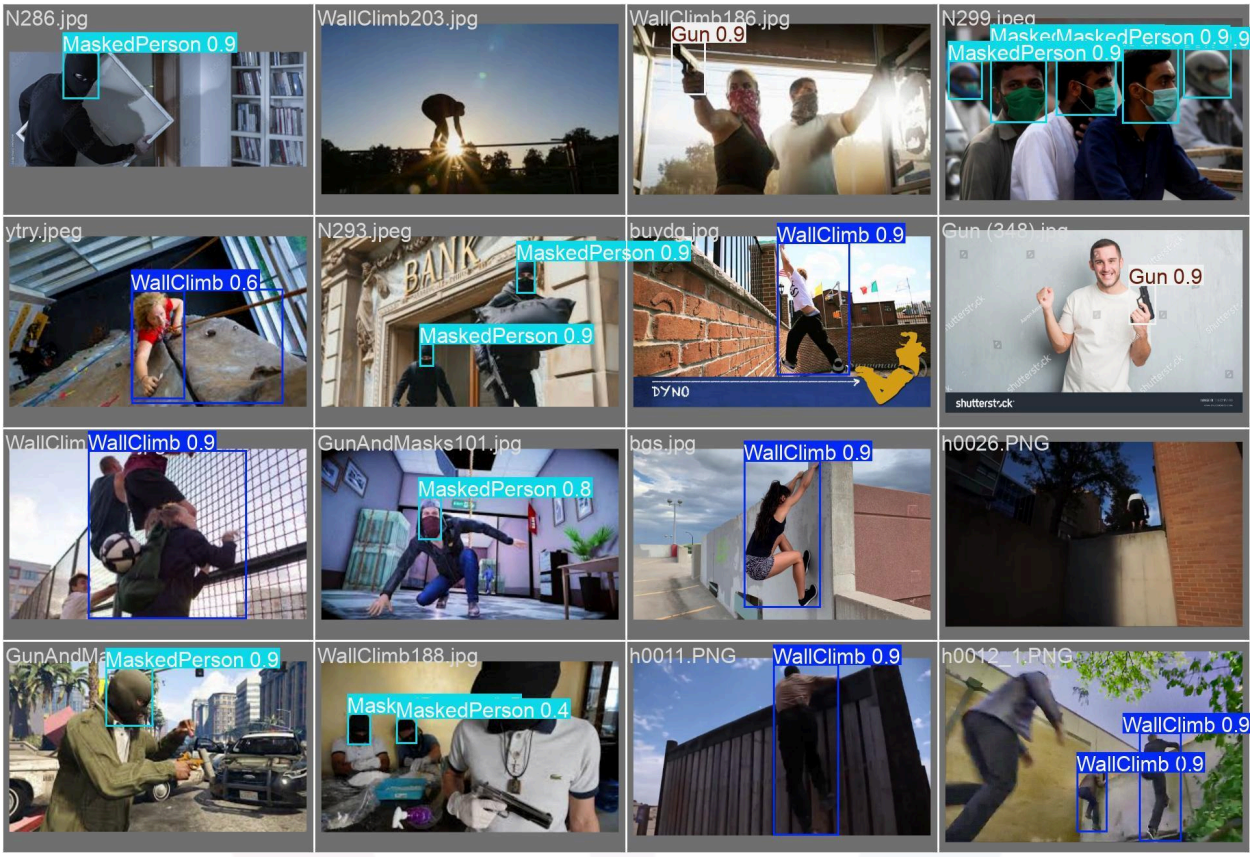
5.6 Training Results







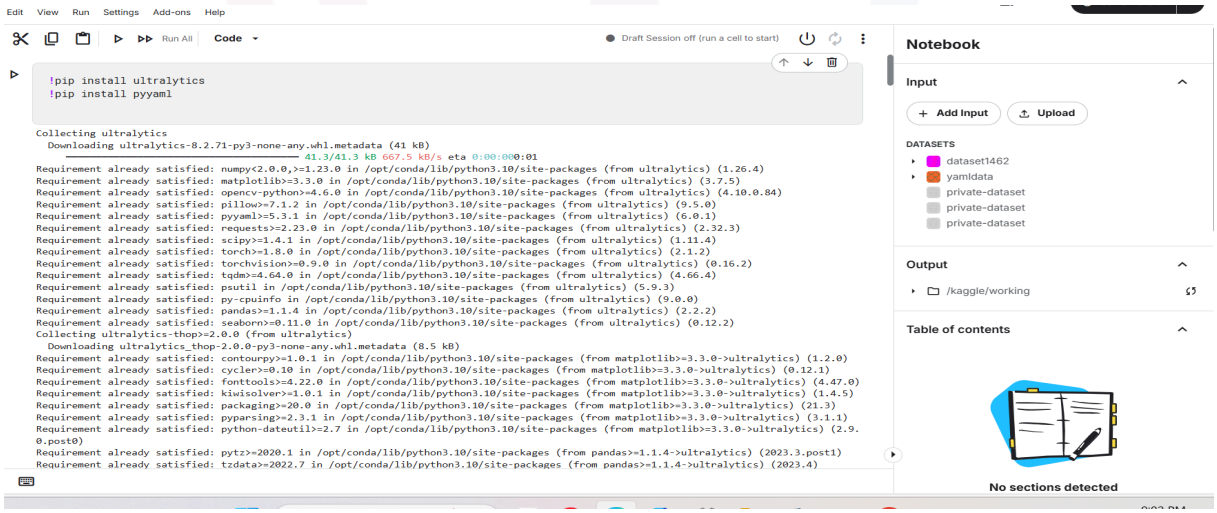


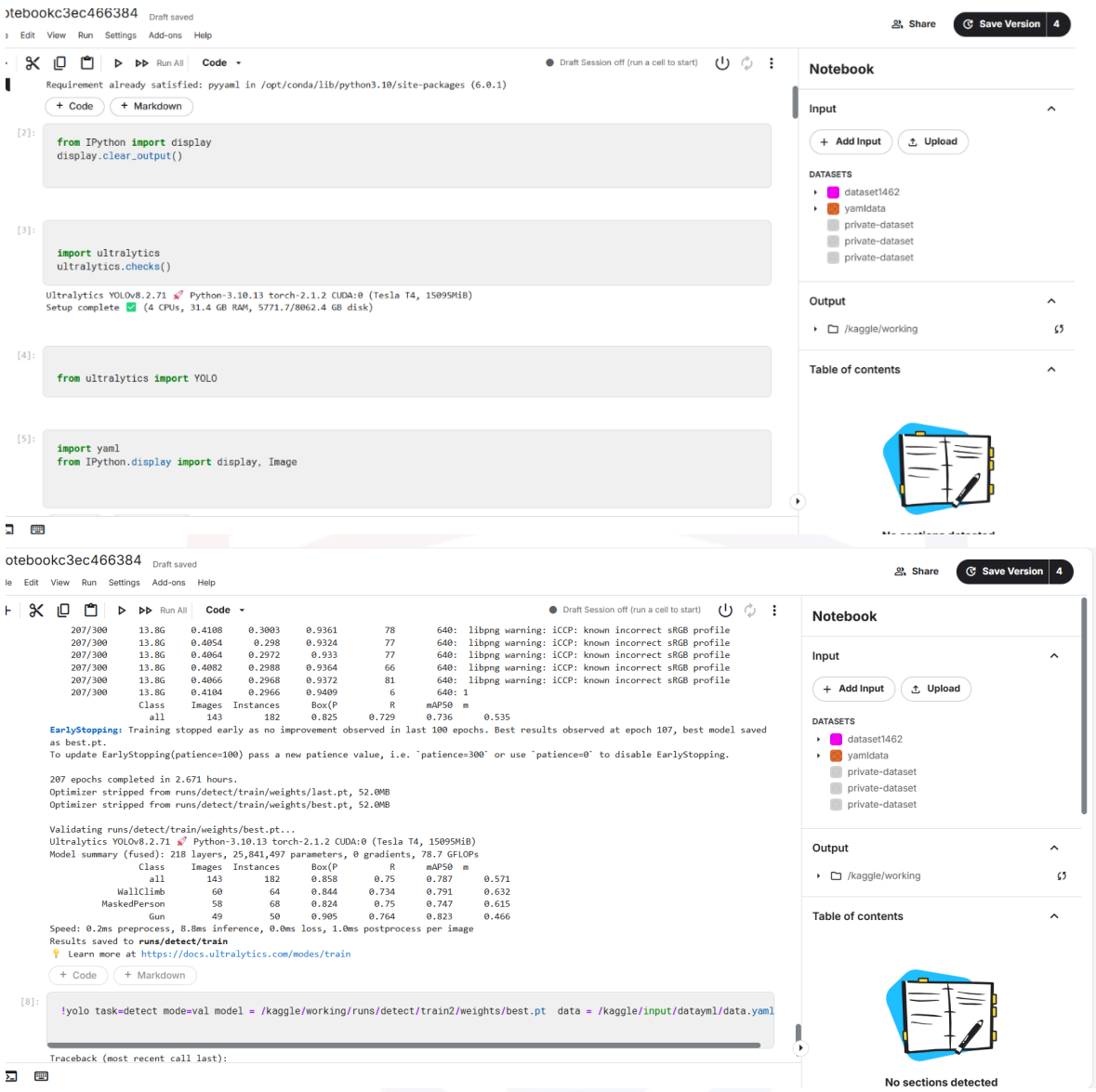

N286.jpg
MaskedPerson 0.9
WallClimb203.jpg
WallClimb186.jpg
Gun 0.9
N299.jpeg
MaskedPerson 0.9
MaskedPerson 0.9
ytry.jpeg
WallClimb 0.6
N293.jpeg
BANK
MaskedPerson 0.9
MaskedPerson 0.9
buydg.jpg
WallClimb 0.9
DYNO
Gun (348).jpg
Gun 0.9
shutterstock
WallClim
WallClimb 0.9
GunAndMasks101.jpg
MaskedPerson 0.8
bgs.jpg
WallClimb 0.9
h0026.PNG
GunAndMa
MaskedPerson 0.9
WallClimb188.jpg
Mask
MaskedPerson 0.4
h0011.PNG
WallClimb 0.9
h0012_1.PNG
WallClimb 0.9
WallClimb 0.9

Other Models

We have trained several models, including ResNet and LSTM-CNN, but they did not perform well in real-time testing. The ResNet model, in particular, exhibited overfitting and struggled to generalize to unseen data. On the other hand, YOLOv8 demonstrated significantly better performance. We trained two YOLOv8 models: one Medium model with 300 epochs and another Large model with 500 epochs. Despite the Large model achieving a high confidence rate, the Medium model proved to be more effective, achieving a notable accuracy of 78% on Kaggle's GPU infrastructure.

In addition to achieving solid accuracy, YOLOv8 excelled in real-time applications. We specifically trained YOLOv8 for several use cases including wall climbing, face covering, and weapon detection using real-time footage from CCTV cameras. This involved extensive training with diverse and challenging real-world data to ensure robustness and accuracy in dynamic environments. The Medium model's superior performance underscores its effectiveness in practical scenarios, where it has shown to consistently deliver reliable results in real-time video analysis.




otebookc3ec466384
Draft saved
Edit View Run Settings Add-ons Help
Share
Save Version 4
Run All
Code
Draft Session off (run a cell to start)
Requirement already satisfied: pyyaml in /opt/conda/lib/python3.10/site-packages (6.0.1)
+ Code + Markdown
[2]:
from IPython import display
display.clear_output()
[3]:
import ultralytics
ultralytics.checks()
Ultralytics YOLOv8.2.71 Python-3.10.13 torch-2.1.2 CUDA:0 (Tesla T4, 15095MiB)
Setup complete (4 CPUs, 31.4 GB RAM, 5771.7/8062.4 GB disk)
[4]:
from ultralytics import YOLO
[5]:
import yaml
from IPython.display import display, Image
Notebook
Input
+ Add Input
Upload
DATASETS
dataset1462
yamldata
private-dataset
private-dataset
private-dataset
Output
/kaggle/working
Table of contents

otebookc3ec466384
Draft saved
Edit View Run Settings Add-ons Help
Share
Save Version 4
Run All
Code
Draft Session off (run a cell to start)
207/300 13.8G 0.4108 0.3003 0.9361 78 640: libpng warning: iCCP: known incorrect sRGB profile
207/300 13.8G 0.4054 0.298 0.9324 77 640: libpng warning: iCCP: known incorrect sRGB profile
207/300 13.8G 0.4064 0.2972 0.933 77 640: libpng warning: iCCP: known incorrect sRGB profile
207/300 13.8G 0.4082 0.2988 0.9364 66 640: libpng warning: iCCP: known incorrect sRGB profile
207/300 13.8G 0.4066 0.2968 0.9372 81 640: libpng warning: iCCP: known incorrect sRGB profile
207/300 13.8G 0.4104 0.2966 0.9409 6 640: 1
Class Images Instances Box(P R mAP50 m
all 143 182 0.825 0.729 0.736 0.535
EarlyStopping: Training stopped early as no improvement observed in last 100 epochs. Best results observed at epoch 107, best model saved as best.pt.
To update EarlyStopping(patience=100) pass a new patience value, i.e. `patience=300` or use `patience=0` to disable EarlyStopping.
207 epochs completed in 2.671 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 52.0MB
Optimizer stripped from runs/detect/train/weights/best.pt, 52.0MB
Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.2.71 Python-3.10.13 torch-2.1.2 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 218 layers, 25,841,497 parameters, 0 gradients, 78.7 GFLOPs
Class Images Instances Box(P R mAP50 m
all 143 182 0.858 0.75 0.787 0.571
Wallclimb 60 64 0.844 0.734 0.791 0.632
MaskedPerson 58 68 0.824 0.75 0.747 0.615
Gun 49 50 0.905 0.764 0.823 0.466
Speed: 0.2ms preprocess, 8.8ms inference, 0.0ms loss, 1.0ms postprocess per image
Results saved to runs/detect/train
Learn more at https://docs.ultralytics.com/modes/train
+ Code + Markdown
[8]:
!yolo task=detect mode=val model = /kaggle/working/runs/detect/train2/weights/best.pt data = /kaggle/input/datayml/data.yaml
Traceback (most recent call last):
Notebook
Input
+ Add Input
Upload
DATASETS
dataset1462
yamldata
private-dataset
private-dataset
private-dataset
Output
/kaggle/working
Table of contents
No sections detected

Challenges

- IP Camera Feed Integration: Required custom solutions to display RTSP streams inside the app.
- Alarm Triggering: Multiple hardware and software solutions were tested before selecting the Node.js server-triggered alarm.
- Telecommunication Restrictions: The inability to send automatic calls or SMS led us to integrate WhatsApp as an alternative.
- Notification Overload: The YOLOv8 model initially sent too many notifications, which we managed by implementing smarter notification control.

Research

In this mission we have review some articles on deep learning which uses for intrusion detection and we gather their ideas and results of their model. For this we get some info about some Machine Learning Algorithms that which are used and what speeds, Accuracy Average in pervious trained models, some of them we mention below:

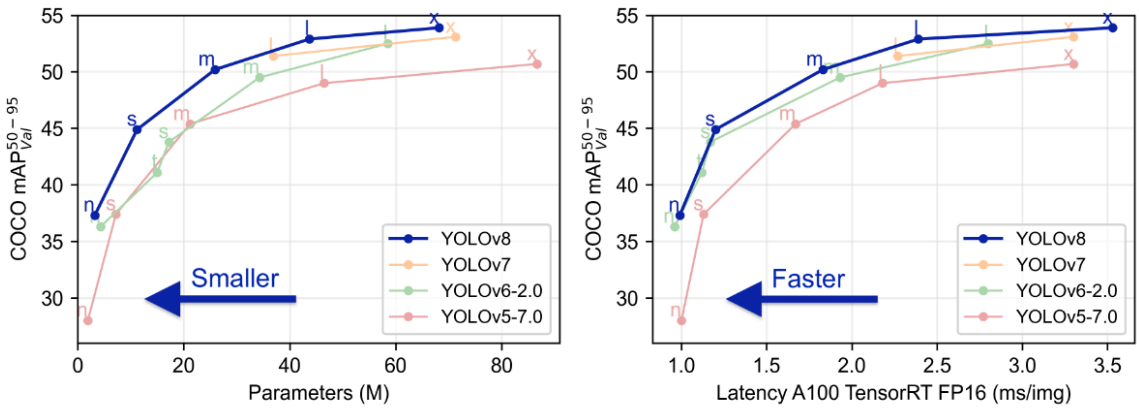
Models	Accuracy	Speed
CNN+LSTM	66%	0.05fps
YOLO	69%	45fps
Fast Yolo	16%	100fps

So because of Speed of YOLO was Fast and the Accuracy was good, we had chosen it and then we searched about it and we found an article that also uses yolo for fighting detection.

Ultralytics YOLOv8 is a cutting-edge, state-of-the-art (SOTA) model that builds upon the success of previous YOLO versions and introduces new features and improvements to further boost performance and flexibility. YOLOv8 is designed to be fast, accurate, and easy to use, making it an excellent choice for a wide range of object detection and tracking, instance segmentation, image classification and pose estimation tasks.

We hope that the resources here will help you get the most out of YOLOv8. Please browse the YOLOv8 Docs for details, raise an issue on GitHub for support, questions, or discussions, become a member of the Ultralytics Discord, Reddit and Forums!

To request an Enterprise License please complete the form at Ultralytics Licensing.



GitHub - ultralytics/ultralytics: NEW - YOLOv8 🚀 in PyTorch > ONNX > OpenVINO > CoreML > TFLite

Paper Name: Suspicious Activity Detection from Videos using YOLOv3

Author names: Nipunjita Bordoloi;Anjan Kumar Talukdar;Kandarpa Kumar Sharma

Abstract: Human activity detection for video system is an automated way of processing video sequences and making an intelligent decision about the actions in the video. It is one of the growing areas in Computer Vision and Artificial Intelligence. Suspicious activity detection is the process of detecting unwanted human activities in places and situations. This is done by converting video into frames and analyzing the activities of persons from the processed frames.

Using visual surveillance, human activities can be monitored in public areas such as bus stations, railway stations, airports, banks, shopping malls, school and colleges, parking lots, roads, etc. to prevent terrorism, accidents and illegal parking, vandalism, fighting, crime and other suspicious activities.

Paper3890.pdf (ijarsct.co.in)

Conclusion

The Emergency Alert App successfully integrates real-time video detection, IoT devices, and communication systems to create a seamless emergency response solution. The combination of machine learning for detecting suspicious activities and IoT alarm systems greatly improves response times in critical situations. While some challenges, such as automatic calls, remain unresolved due to external factors, the app achieves its core goal of enhancing user safety and security.

References

[1]. *Elizabeth Scaria, Aby Abahai T and Elizabeth Isaac, "Suspicious Activity Detection in Surveillance Video using Discriminative Deep Belief Network", International Journal of Control Theory and Applications Volume 10, Number 29 -2017.*

[2]. *Singh P and Pankajakshan V 2018 A Deep Learning Based Technique For Anomaly Detection In Surveillance Videos. Proc. of the 24th National Conf. on Communications, pp. 1-6.*

[3]. *Joey Tianyi Zhou, Jiawei Du, Hongyuan Zhu, Xi Peng, Rick Siow Mong Goh, "AnomalyNet: An Anomaly Detection Network for Video Surveillance, 2019.*

[4]. *Monika D. Rokade and Tejashri S. Bora, "Survey On Anomaly Detection for Video Surveillance" 2021 International Research Journal of Engineering and Technology(IRJET).*

[5]. P.Bhagya Divya, S.Shalini, R.Deepa, Baddeli Sravya Reddy,"Inspection of suspicious human activity in the crowdsourced areas captured in surveillance cameras",International Research Journal of Engineering and Technology (IRJET), December 2017.