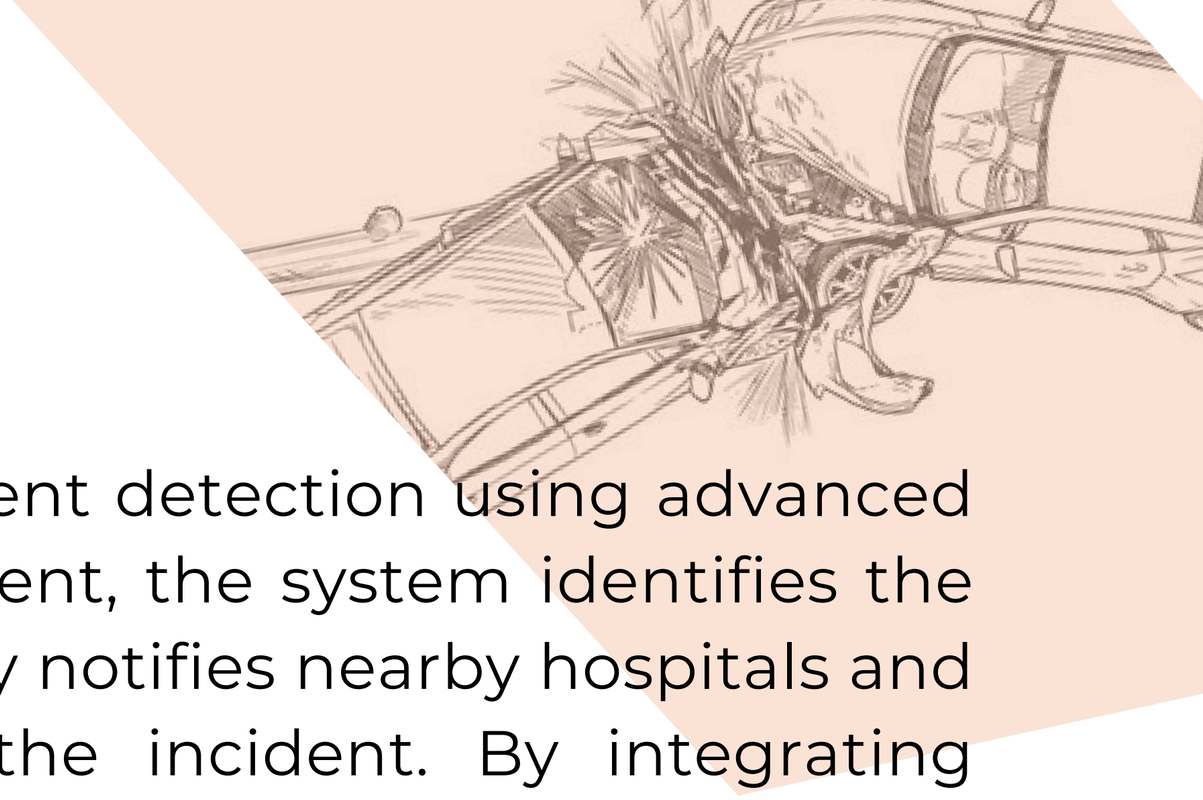
Major Project

Accident Detection

(Alert System)

N. Sai Sriram
V. Sai Lokesh
B. Ajith Kumar
Sk. Alisha
K. Lokesh Krishna

Table of Contents



ABSTRACT

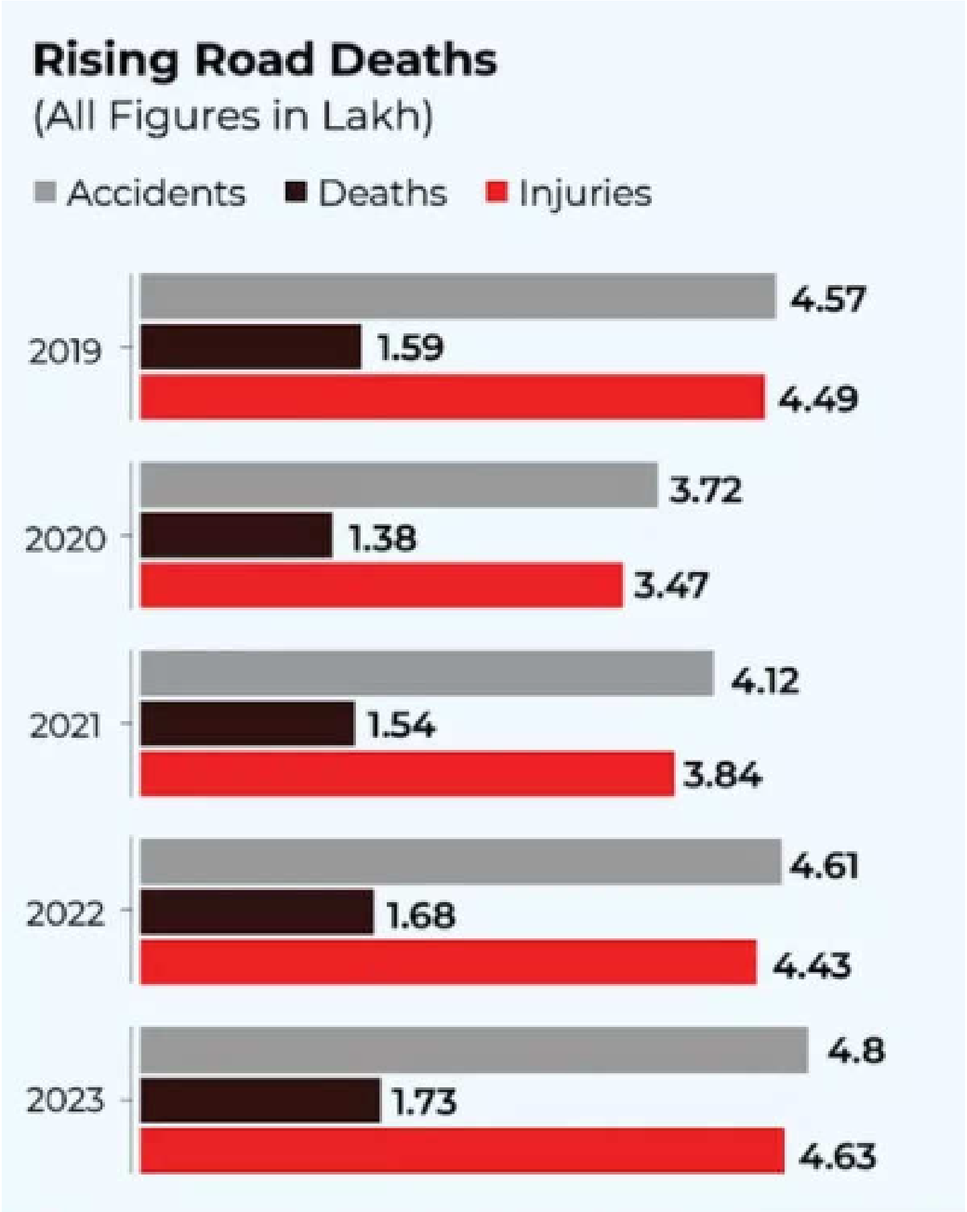
This project focuses on real-time accident detection using advanced technologies. Upon detecting an accident, the system identifies the vehicle’s number plate and immediately notifies nearby hospitals and the vehicle owner's relatives about the incident. By integrating accident detection with number plate recognition and automated alert systems, this solution aims to provide timely assistance, potentially saving lives and reducing emergency response times.

Introduction

Accidents are a major concern worldwide, causing loss of life and severe injuries. Quick action after an accident can save lives, but delays in identifying the incident and informing emergency services often worsen the situation. This motivated us to develop a system that can detect accidents in real time, identify the vehicle involved, and send alerts to the necessary people and nearby hospitals.

Motivation for the work

Over the past few years, road fatalities in India have seen a sharp rise—from 1.38 lakh in 2020 to 1.73 lakh in 2023, marking a 25% increase. In 2023 alone, there were 4.8 lakh accidents and 4.63 lakh injuries, highlighting the high severity and frequency of road mishaps. These alarming figures underscore the urgent need for a real-time accident detection and alert system that can reduce emergency response delays and potentially save countless lives.



Real-World Applications:

Automated alerts to hospitals and family members can help save lives.

Real-time accident detection can reduce emergency response times.

It can be integrated into smart city solutions to improve road safety.

Related Works

- Existing accident detection systems rely on sensors or image processing but often lack real-time accuracy and automated alerting capabilities.

- Number plate recognition technologies are widely used but are rarely integrated into accident response systems.

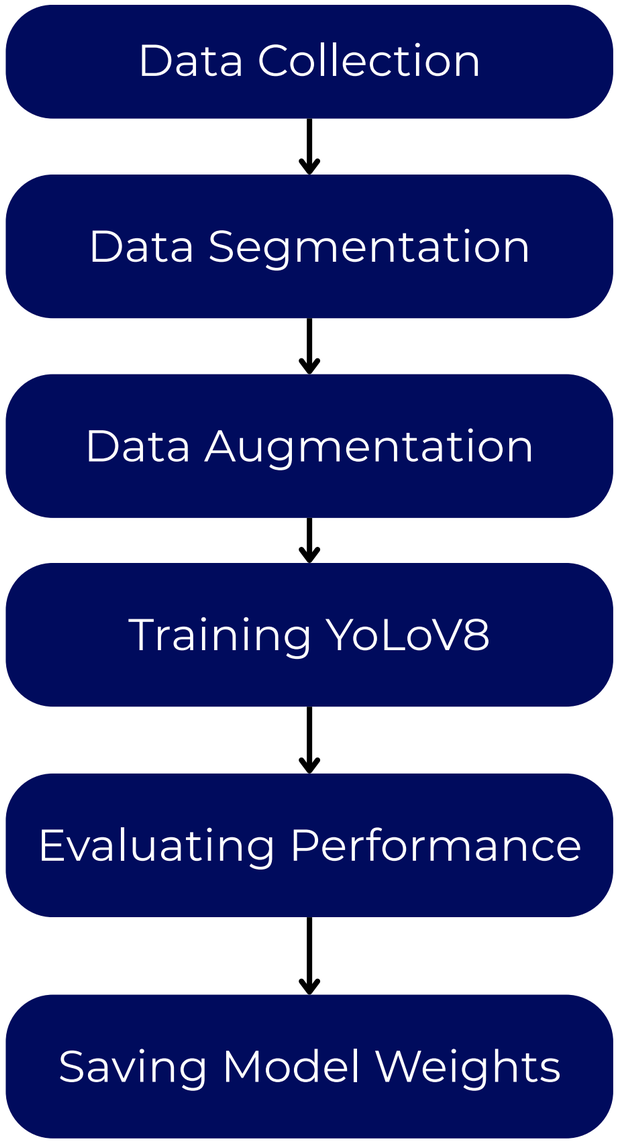
- Many emergency response solutions depend on manual reporting, leading to delays in assistance

- Current systems fail to provide a comprehensive solution combining accident detection, number plate recognition, and automated notifications to hospitals and relatives

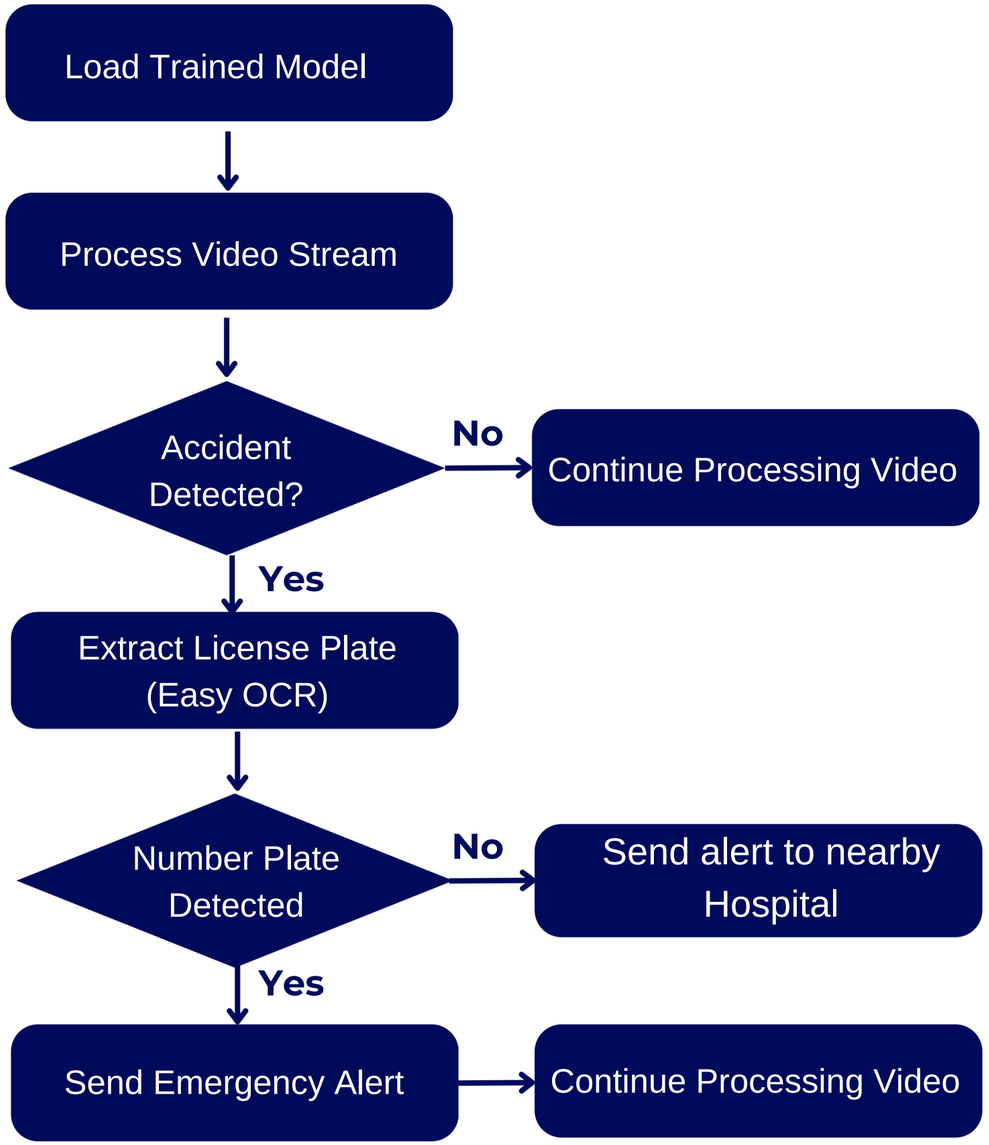
Literature Overview

- Railway systems use automated signals to quickly notify about accidents, ensuring immediate response.
- On-road accidents lack such real-time alert mechanisms, causing delays in emergency assistance.
- Our project addresses this by providing real-time accident detection and automated alerts to hospitals and relatives, improving response times and saving lives.

Training Flowchart :



Proposed Method



Technical details

- **Programming Language:** Python
- **Frameworks:** Torch
- **Libraries: OCR:** EasyOCR
- **Notifications:** Twilio API
- **Deployment :** Django
- **Model:** Trained : YoLo V8 (for accident detection)
- **Training:** Binary classification (Accident/No Accident)
- **Hardware:** GPU support (optional for scalability) Minimum 8 GB RAM for local testing.

Pseudo Code for Training

1. Dataset Collection

a. Gather accident-related images.
b. Annotate images with bounding boxes in YOLO format.

2. Data Segmentation

a. Split dataset into training, validation, and test sets.

3. Initialize Data Augmentation

a. Apply brightness adjustment to simulate lighting conditions.
b. Apply horizontal and vertical flips for spatial variance.
c. Adjust contrast to enhance object distinction.

4. Load Pre-trained YOLOv8 Model

a. Use yolov8s.pt for a lightweight, fast model.
b. Load model with pre-trained weights for better convergence.

Pseudo Code for Training

5. Train the Model

a. Configure training parameters:

- Epochs = 50
- Image size = 512
- Batch size = 32
- AMP (mixed precision) enabled for speed

b. Use GPU (CUDA) for faster computation.

c. Save model checkpoints after each epoch.

6. Evaluate the Model

a. Evaluate performance using the validation and test datasets.

b. Analyze metrics: Precision, Recall, mAP.

c. Save the final trained model weights.

Pseudo Code for Model Deployment

1. Upload Video via Web Interface

a. User uploads a video through the Django-based web page.
b. Video is saved on the server for processing.

2. Run Accident Detection Using YOLOv8

a. Extracting frames from the uploaded video.
b. Passing each frame to the trained YOLOv8 model.
C. If accident is detected in any frame, proceed to the next step.

3. License Plate Extraction

a. Using a EasyOCR to detect and extract the license plate from the accident frame.
b. If a license plate is detected, retrieve the associated phone number from the database.

Pseudo Code for Model Deployment

4. Sending Alert to Vehicle Owner

a. Composing an alert message containing accident details and location.

b. Sending the alert message to the nearby hospital and the relative’s of the vehicle owner.

5. Handling Missing License Plate

a. If no license plate is detected, identifying nearby hospitals using location data.

b. Sending emergency alerts to the nearest hospitals with accident details and coordinates

.

6. Extracting Location Information

a. Using GPS metadata from the video or frame to determine accident location.

b. Convert GPS coordinates to a human-readable address .

7. Displaying Results on Web Page

a. Show accident detection status to the user.

Why We Used YOLOv8 for Accident Detection

- **Real-Time Detection**
→ YOLOv8 is very fast — it can detect accidents instantly from live video or camera feed.

- **High Accuracy**
→ It gives accurate results by identifying accidents with high confidence, even in busy scenes.

- **Lightweight and Efficient**
→ YOLOv8 is optimized to run on normal GPUs and even edge devices (like Jetson Nano or laptops).

- **Anchor-Free Design**
→ Unlike older YOLO versions, YOLOv8 doesn’t need predefined box sizes, so it's better at detecting objects of different shapes and sizes — like crashed vehicles or fallen bikes.

- **Easy to Train with Custom Data**
→ We can train YOLOv8 on our own accident images easily using the Ultralytics library.

YOLOv8 Architecture

1. Input Layer

- The model starts by taking an image as input — for example, a 512x512 pixel photo from a camera.
- Before sending it to the model, the image is resized and normalized (pixel values scaled) to make processing easier and faster.

2. Backbone

- This is the part of the model that learns what's in the image.
- It looks for visual patterns like edges, shapes, and textures — for example, the shape of a car, a fallen bike, or a person.
- It uses special layers:
  - Convolution layers to scan for patterns.
  - Bottleneck layers to compress information while keeping important details.
  - SiLU activation to help the model learn complex patterns more effectively.

3. Neck

- The neck helps the model understand both small and large details from the image.
- It uses structures like **FPN (Feature Pyramid Network)** and **PAN (Path Aggregation Network).**
- These help the model combine features from different layers — so it can detect a small bike crash and a large truck accident in the same image.

4. Head

- This part is responsible for making predictions.
- For every part of the image, it predicts:
  - The bounding box (where the object is located).
  - A confidence score (how sure the model is that an object is present).
  - The class label (what type of object — e.g., “accident”).
- YOLOv8 is anchor-free, which means it doesn’t rely on predefined box sizes. This makes it more flexible and accurate, especially for unusual shapes like accident scenes.

5. Output

- Finally, the model gives a list of all detected objects.
- Each object has:
  - Box coordinates (where it is in the image),
  - Confidence level, and
  - The class name (e.g., “accident”).

Model Training

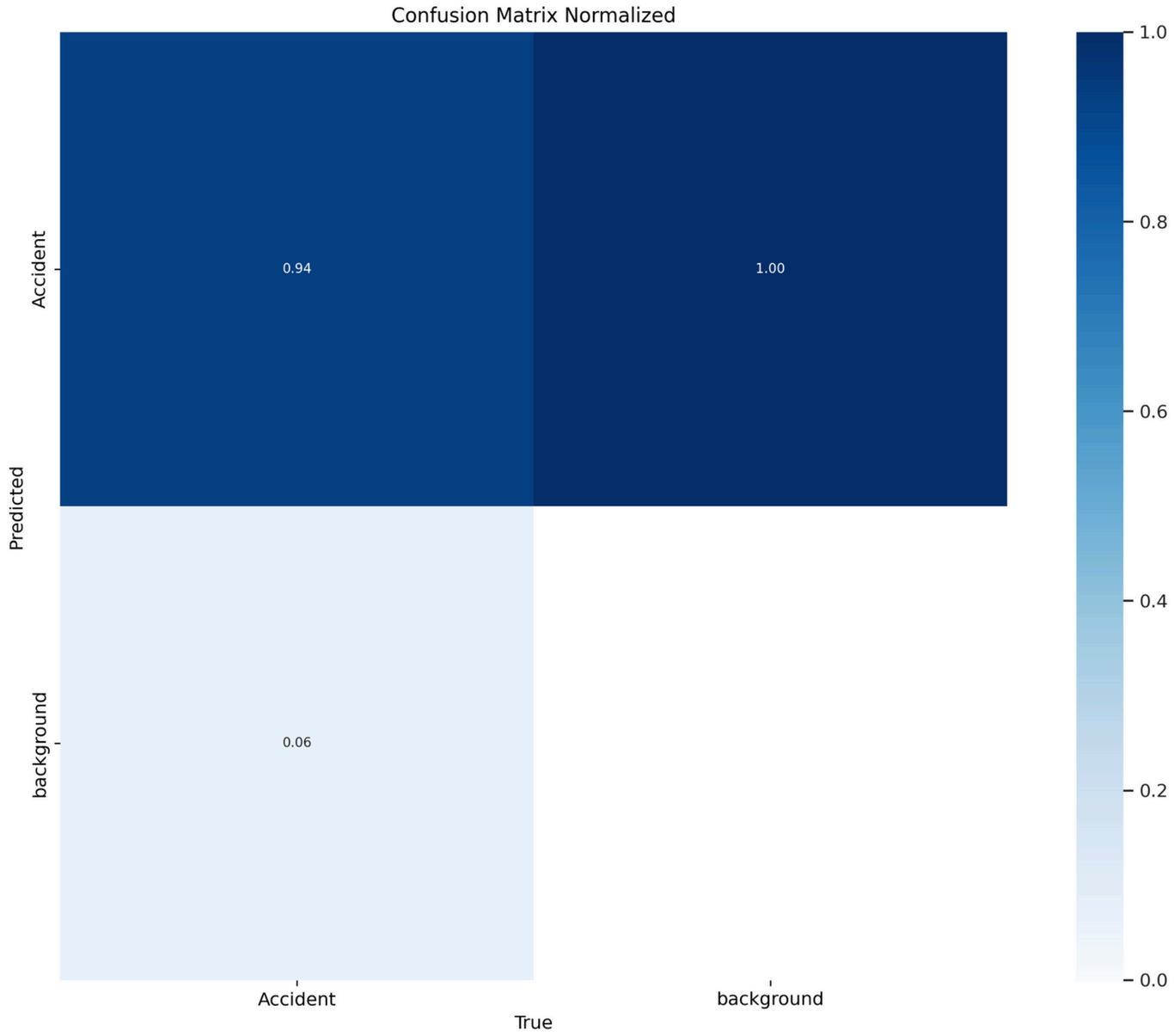
```
from ultralytics import YOLO
import torch

model = YOLO("yolov8s.pt")  # Smaller, faster model

model.train(
    data=data_yaml_path,
    epochs=50,
    imgsz=512,    # smaller images
    batch=32,     # bigger batch if possible
    device="cuda",
    workers=2,
    project='/content/drive/MyDrive/Colab Notebooks/training_results',
    name='accident_model',
    save=True,
    save_period=1,
    cache=True,
    patience=7,
    amp=True,     # mixed precision for speedup
)
```

Model Results

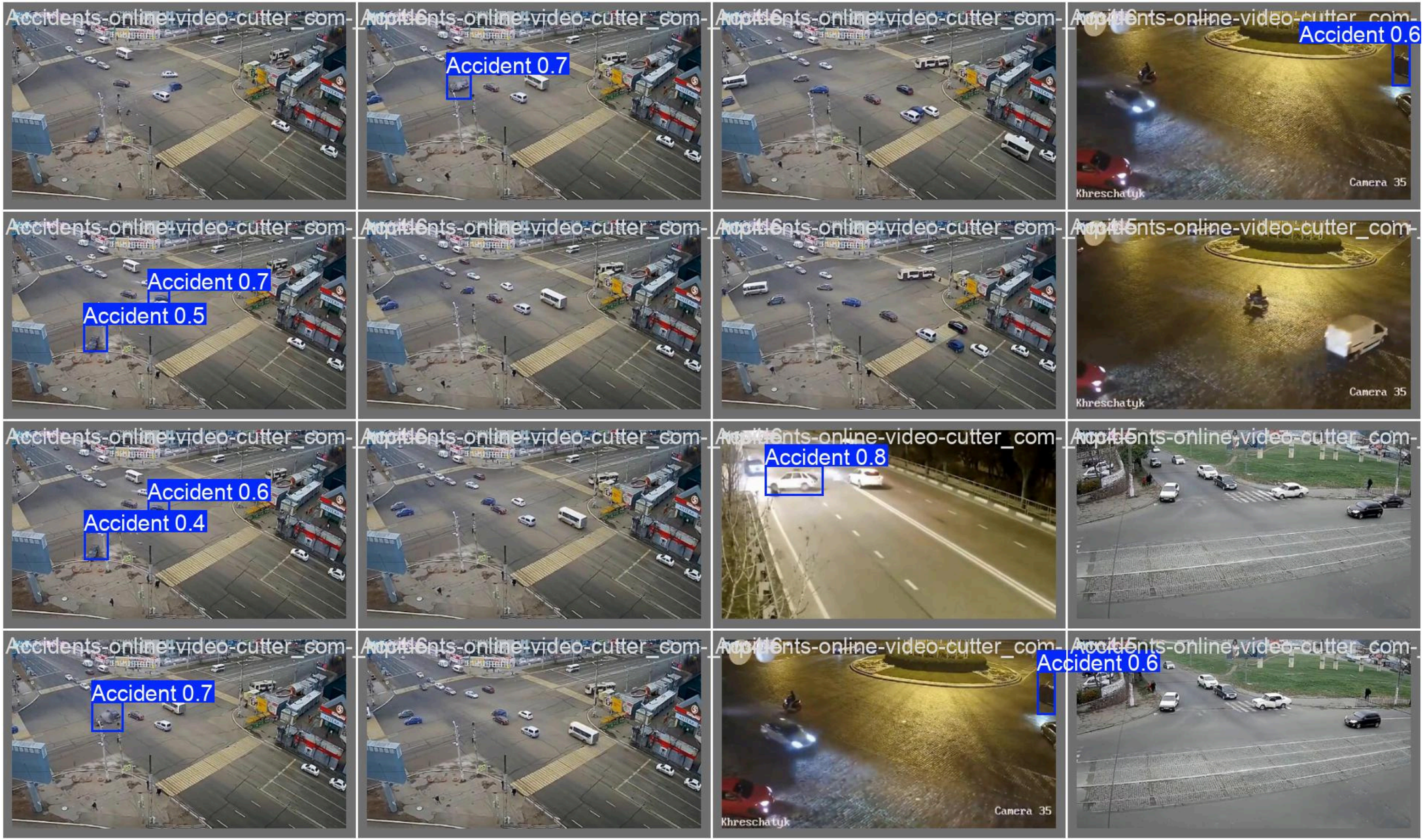
```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
28/50      9.95G     0.8522     0.5515      1.153          1        512: 100%|██████████| 643/643 [04:34<00:00,  2.35it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 24/24 [00:09<00:00,  2.59it/s]                  all       1514       1484      0.95

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
29/50      9.99G     0.8354     0.5373      1.146          3        512: 100%|██████████| 643/643 [04:33<00:00,  2.35it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 24/24 [00:10<00:00,  2.37it/s]                  all       1514       1484      0.959

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
30/50      10.1G      0.831     0.5307      1.144          1        512: 100%|██████████| 643/643 [04:32<00:00,  2.36it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 24/24 [00:10<00:00,  2.36it/s]                  all       1514       1484      0.951

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
31/50      9.99G     0.8177     0.5187      1.134          4        512: 100%|██████████| 643/643 [04:28<00:00,  2.39it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 24/24 [00:10<00:00,  2.33it/s]                  all       1514       1484      0.951

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
32/50      10.1G     0.8085     0.5167      1.133          1        512: 100%|██████████| 643/643 [04:29<00:00,  2.38it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 24/24 [00:09<00:00,  2.47it/s]                  all       1514       1484      0.953
```

Model Evaluation

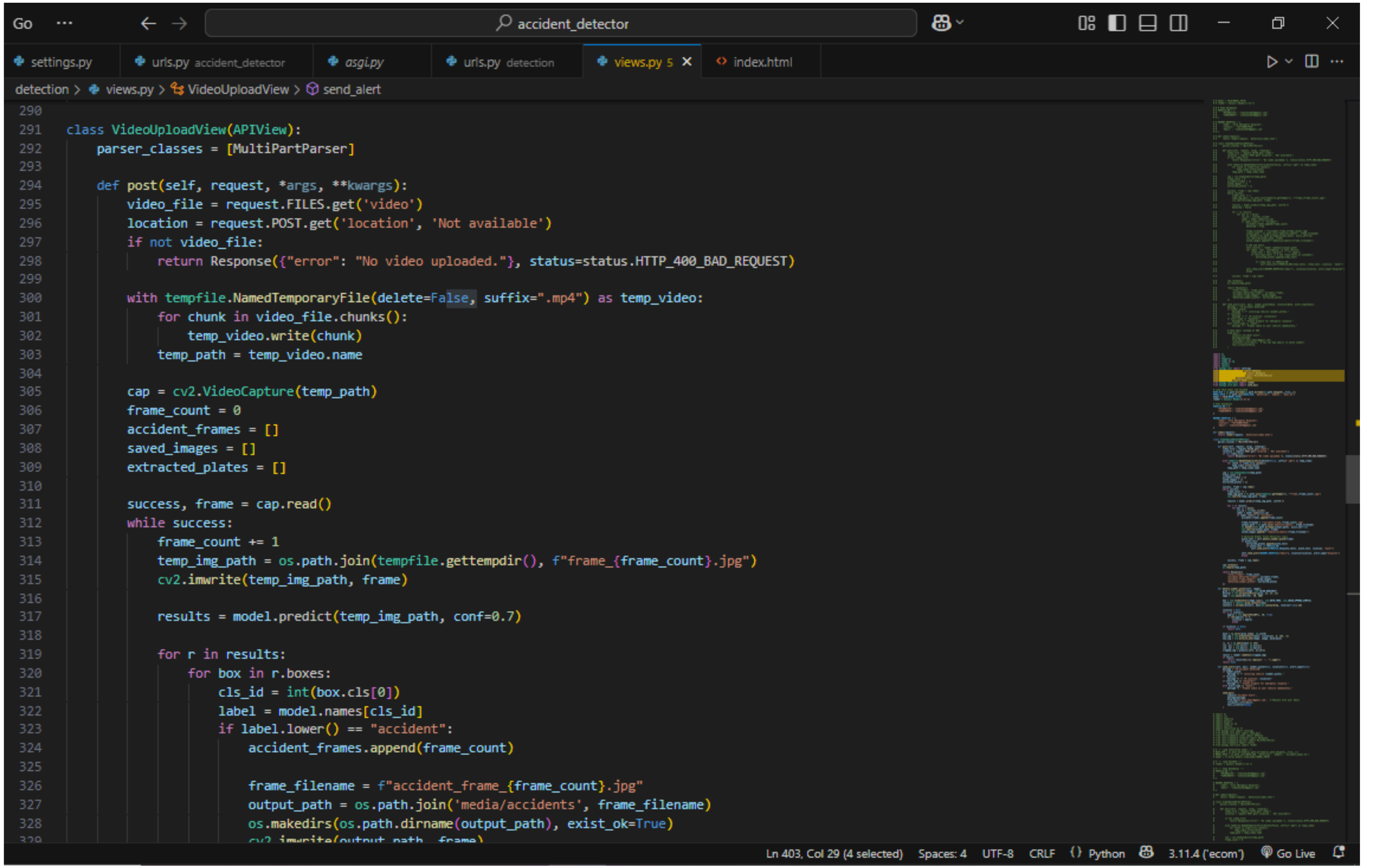
```python
from ultralytics import YOLO

# Load your trained model
model = YOLO("/content/drive/MyDrive/Colab Notebooks/training_results/accidentmodel_22/weights/best.pt")

# Run validation
metrics = model.val(
    data="/content/accident_dataset/dataset/data.yaml",  # your dataset config file
    conf=0.5,  # confidence threshold for evaluation
    split='test'  # important if you want to evaluate on test images
)

# After validation, 'metrics' will have all details
print(metrics)
```


Confusion Matrix Normalized
Accident
background
Predicted
0.94
1.00
0.06
Accident
background
True
1.0
0.8
0.6
0.4
0.2
0.0

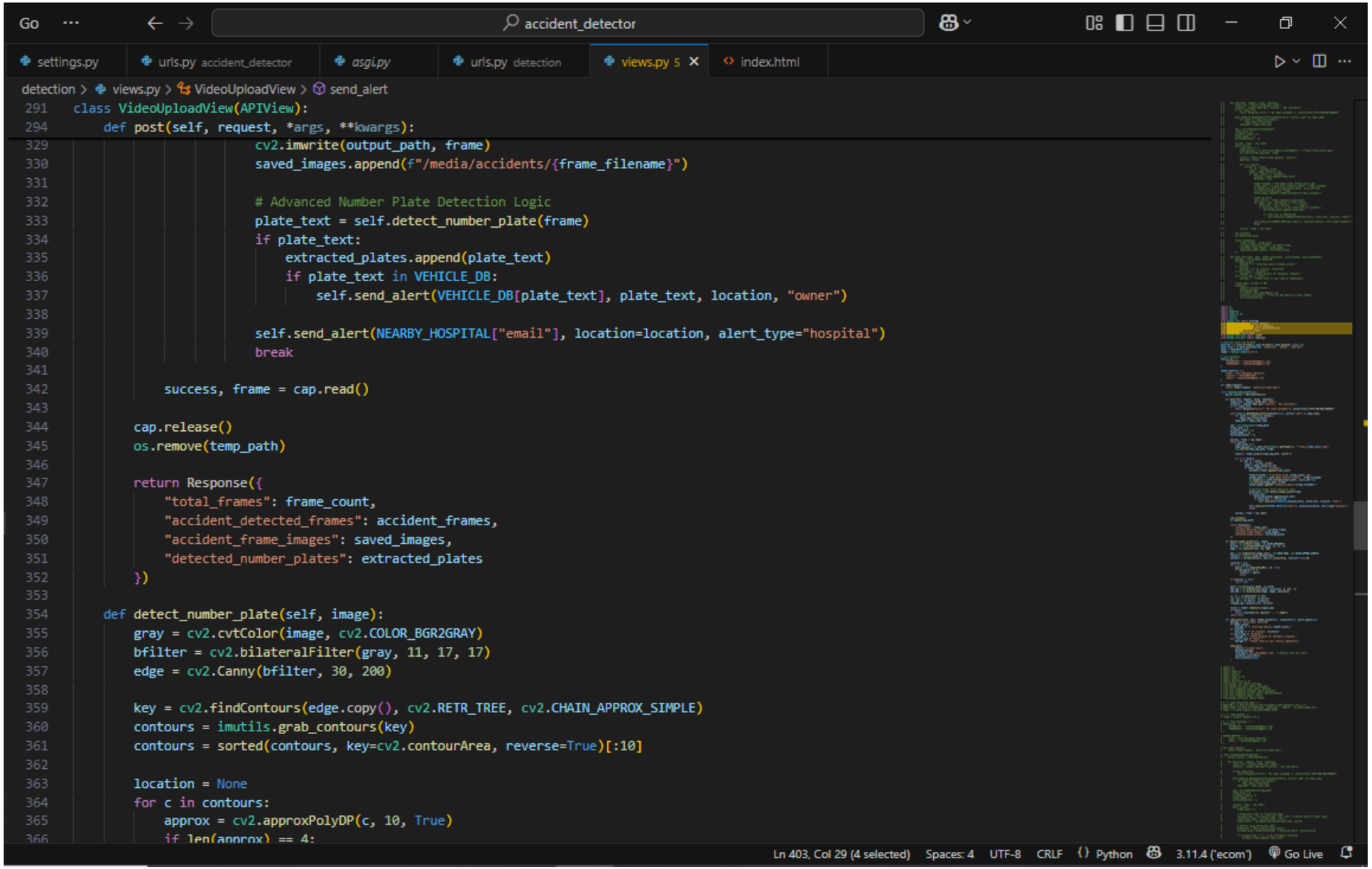

Accidents-online-video-cutter_com-
Accident
Accident


Accident 0.7
Accident 0.6
Camera 35
Khreschatyk
Accident 0.7
Accident 0.5
Camera 35
Khreschatyk
Accident 0.6
Accident 0.4
Accident 0.8
Accident 0.7
Accident 0.6
Camera 35
Khreschatyk

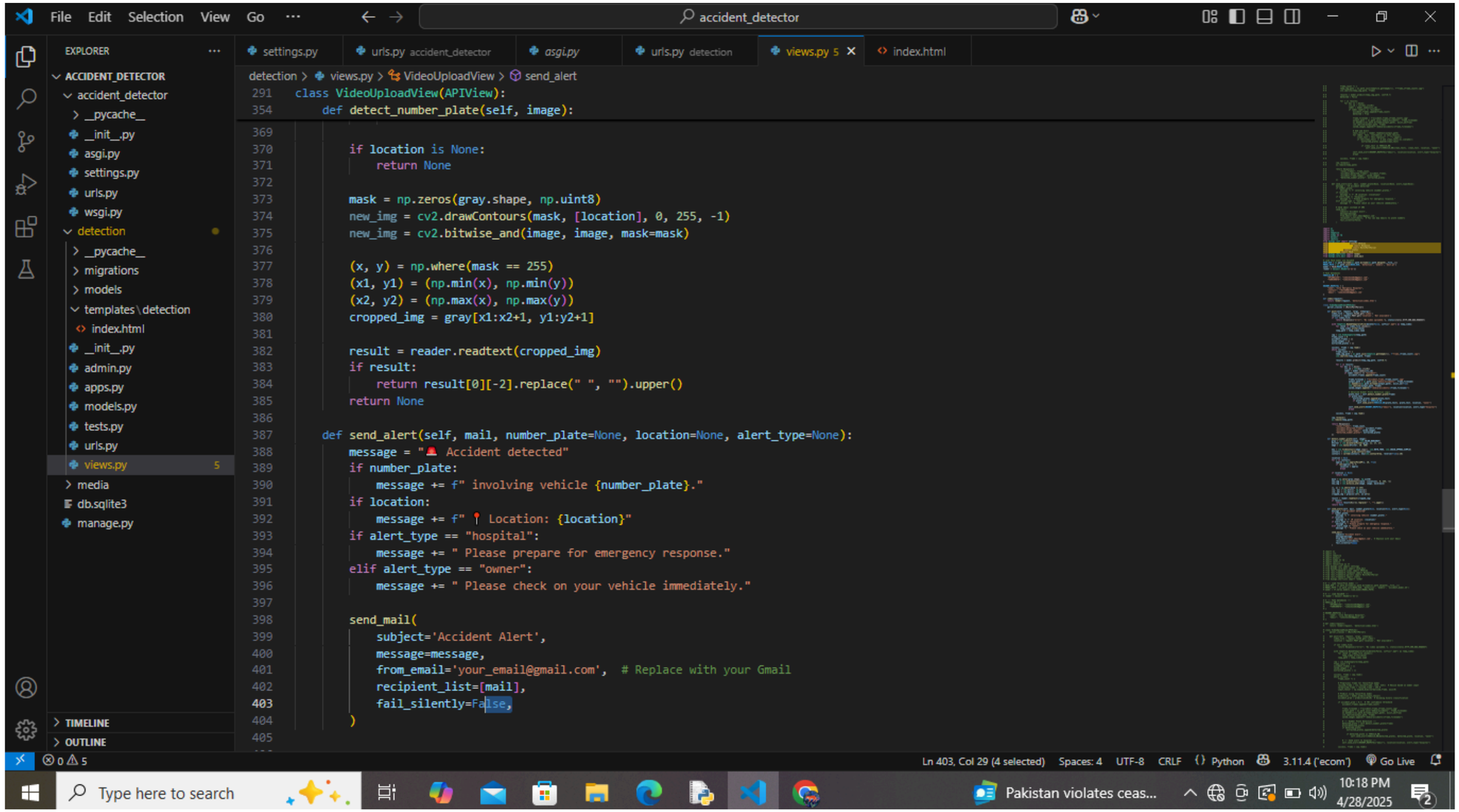
Model Deployment



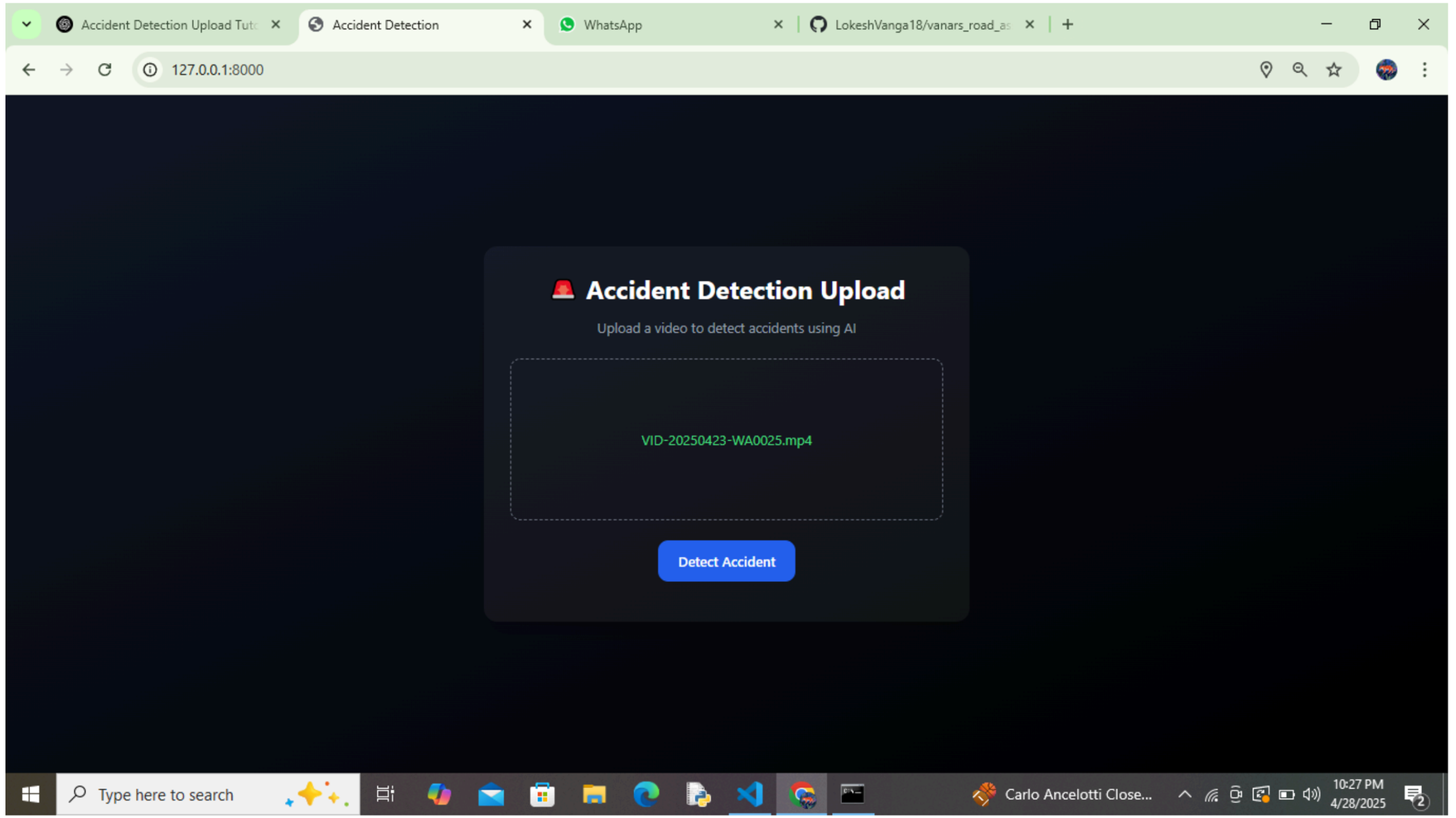
Model Deployment



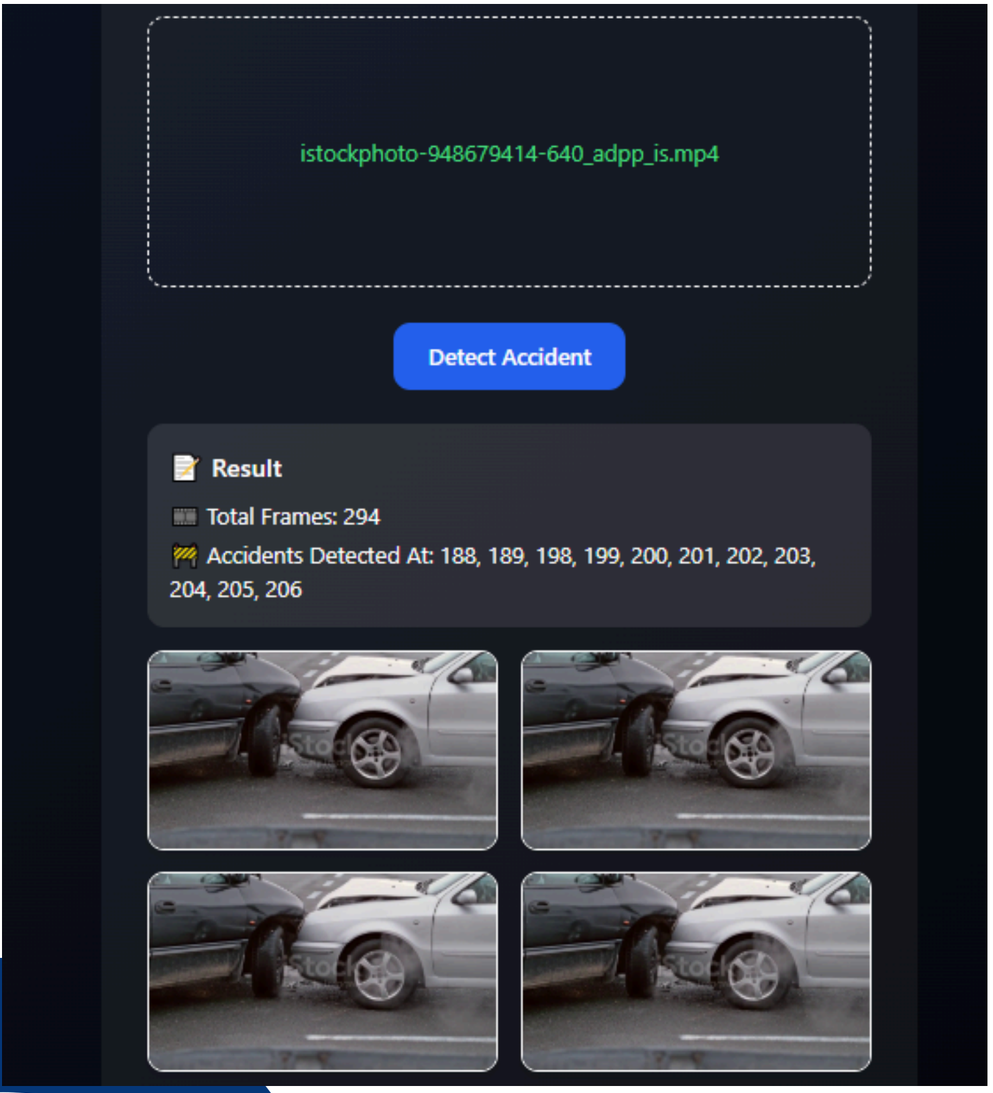
Model Deployment

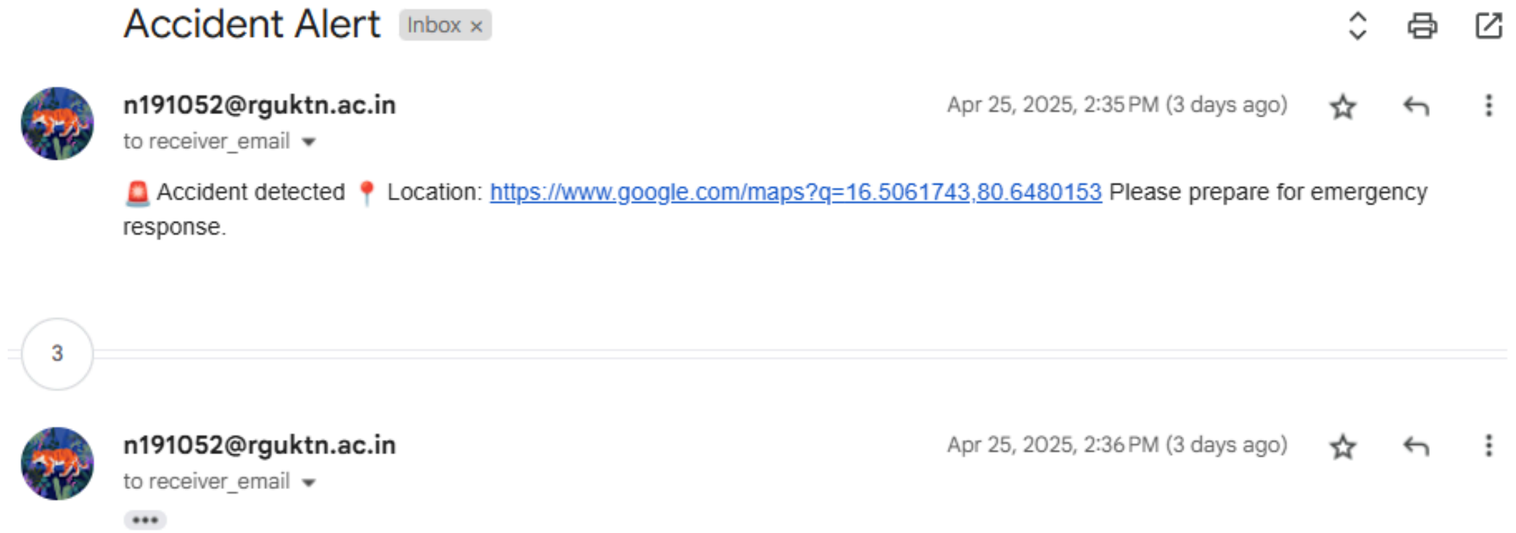


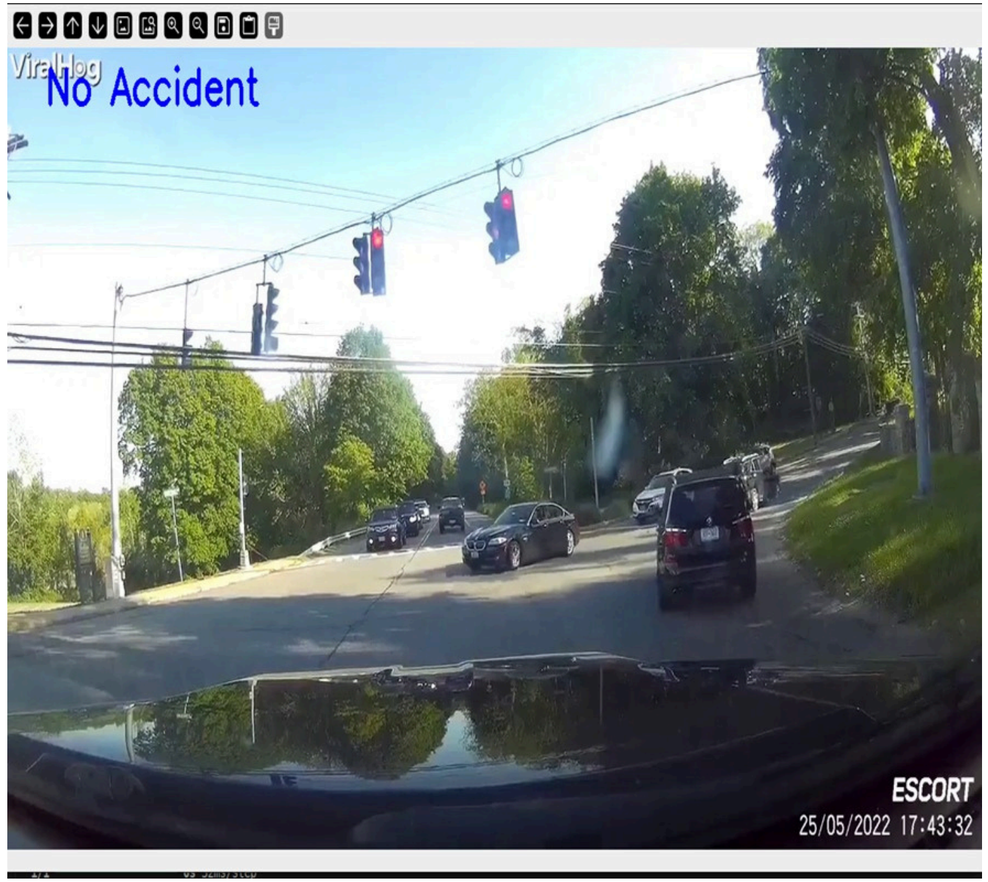
Web Page Interface:

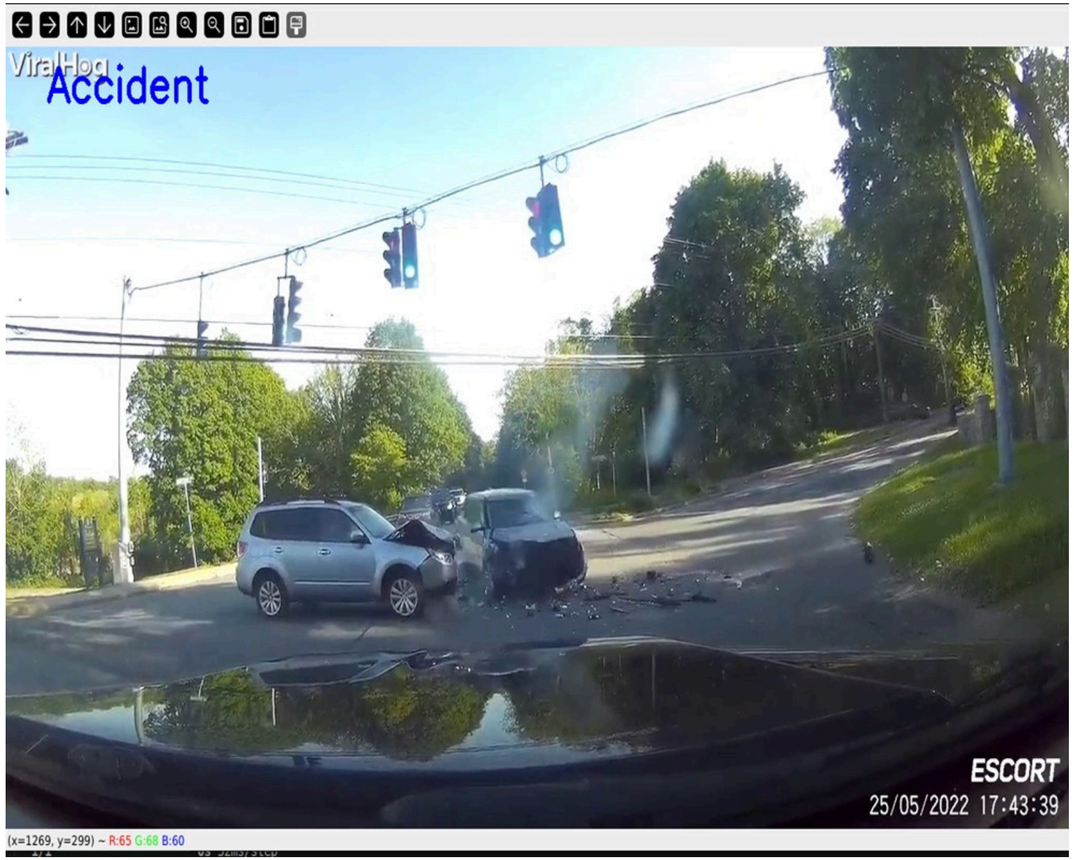


Initial results:






ViralHog
No Accident
ESCORT
25/05/2022 17:43:32


ViralHog
Accident
ESCORT
25/05/2022 17:43:39
(x=1269, y=299) ~ R:65 G:68 B:60

Future Scope :

1. **Enhanced Accuracy:** Improve accident detection and number plate recognition using advanced AI techniques.
2. **Integration with IoT:** Connect with IoT devices for real-time traffic data and better situational awareness.
3. **Multi-language Support:** Develop systems to handle number plates and alerts in multiple languages for global applicability.
4. **Scalability**: Expand the system to work on a large scale, covering diverse environments and traffic conditions.
5. **Preventive Measures:** Use predictive analytics to identify accident-prone zones and propose preventive solutions.

Conclusion

This project introduces an effective way to detect accidents in real-time, recognize vehicle number plates, and send alerts automatically to nearby hospitals and the family members of the victim. By addressing delays in emergency response, it can help save lives and improve road safety. The system brings together advanced technologies to ensure faster and more reliable assistance during critical situations. It lays the groundwork for smarter road safety measures and better accident management in the future.

Reference:

1.Real-Time Accident Detection in Traffic Surveillance Using Deep Learning This study focuses on detecting traffic accidents at intersections using deep learning models. ps://ar5iv.org/abs/2208.06461).

2.Number Plate Recognition (NPR): Number plate recognition is widely used in security and surveillance. https://ijcttjournal.org/2023/Volume-71%20Issue-1/IJCTT-V71I1P102.pdf

Thank You!