



Accident Detection

(Alert System)

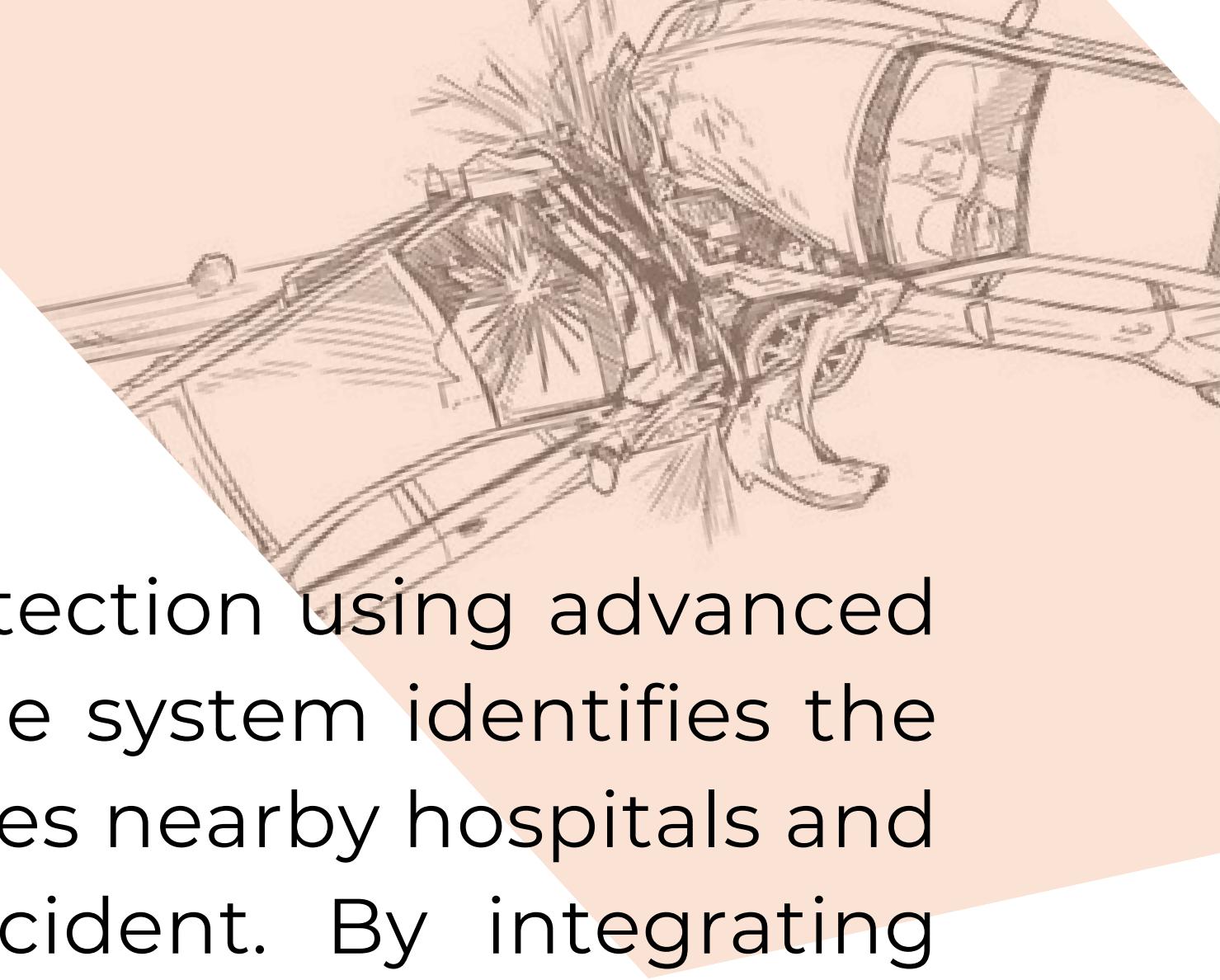
**N. Sai Sriram
v. Sai Lokesh
B. Ajith Kumar
Sk. Alisha
K. Lokesh Krishna**

Table of Contents

- **Abstract**
- **Introduction**
- **Motivation for the work**
- **Real world Applications**
- **Related Works**
- **Literature Overview**
- **Proposed Method**
- **Technical Details**
- **Model Architecture**
- **Future Scope**
- **Conclusion**

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.

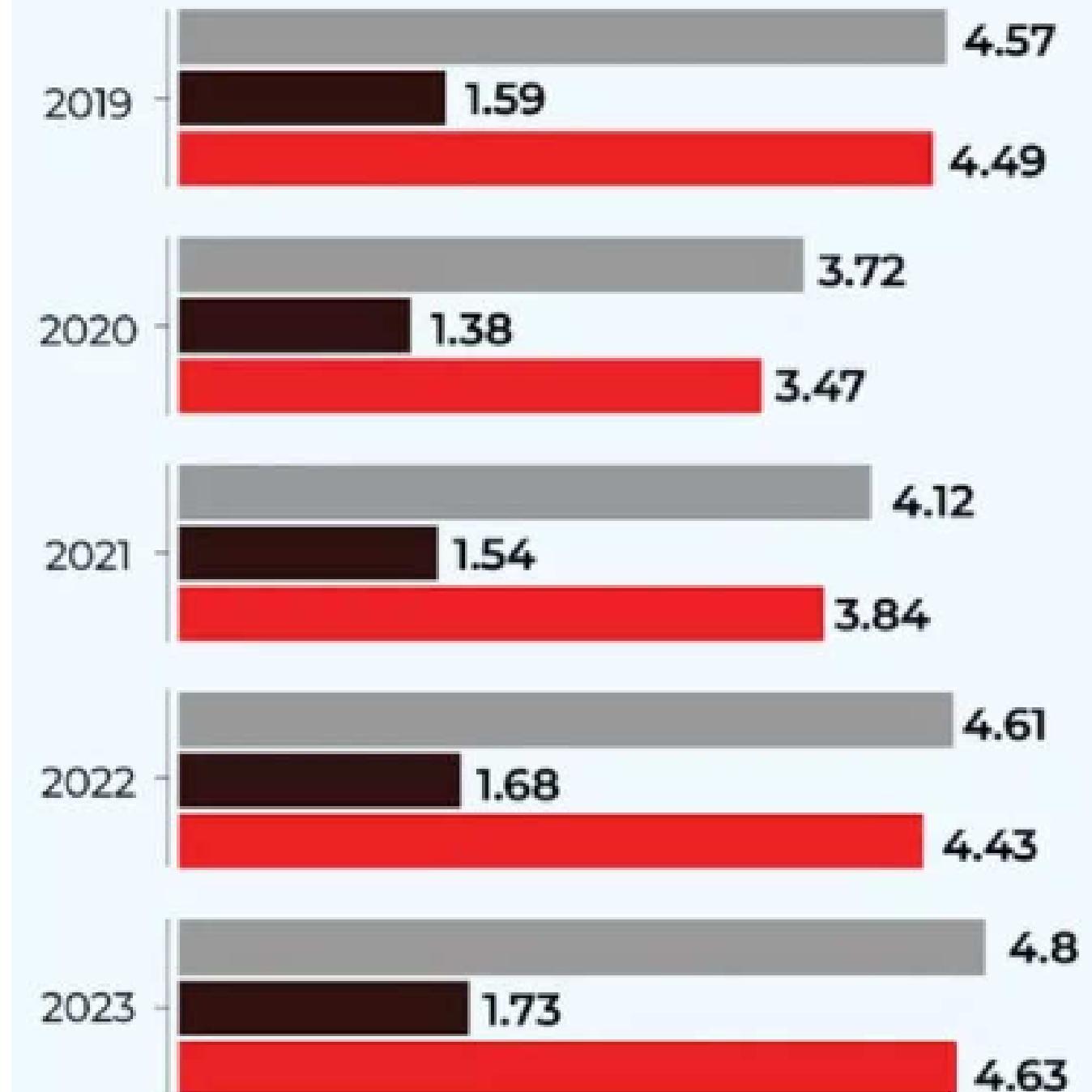
Motivation for the work

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.

Rising Road Deaths

(All Figures in Lakh)

■ Accidents ■ Deaths ■ Injuries



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

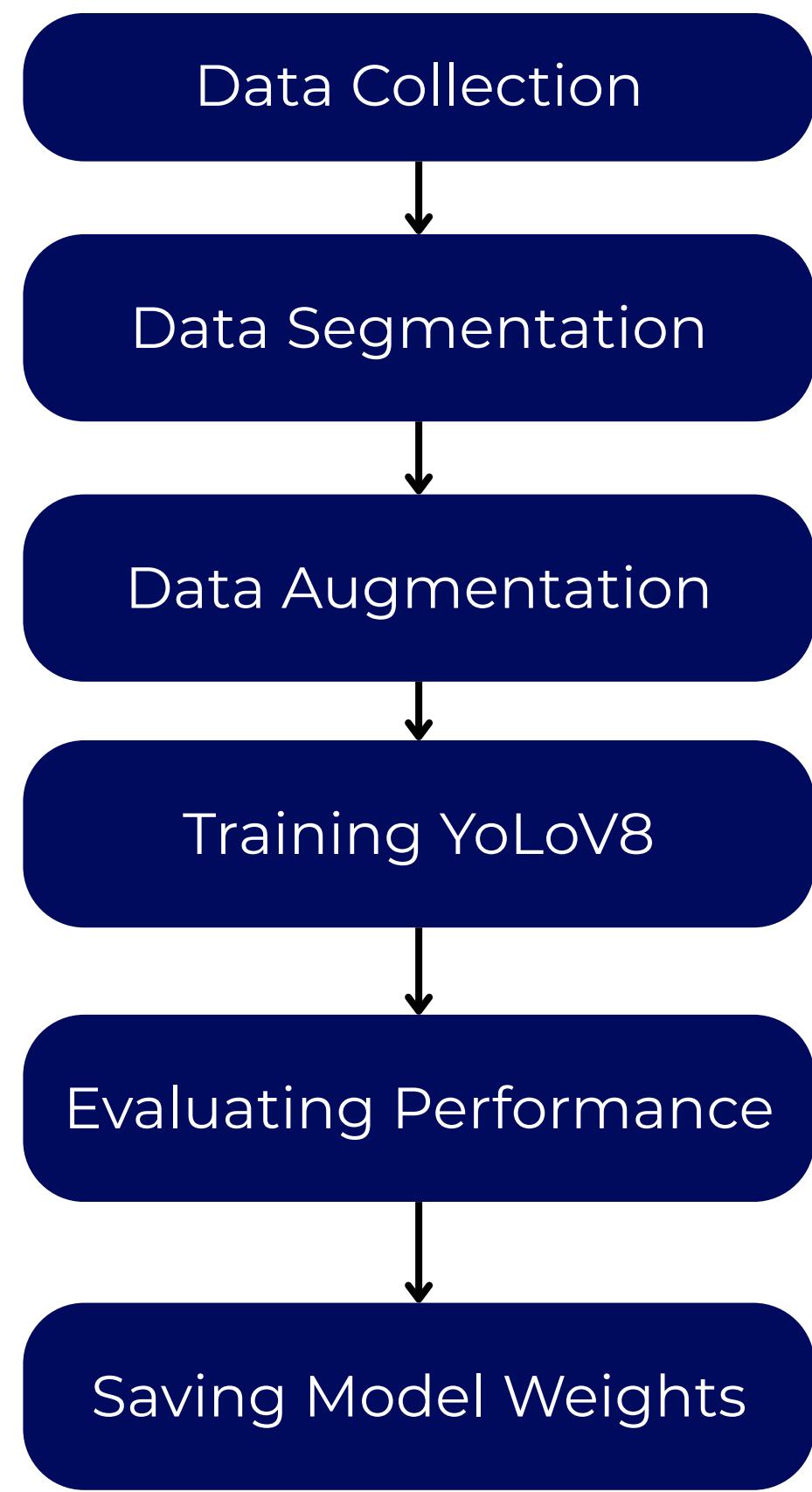
Current systems fail to provide a comprehensive solution combining

- accident detection, number plate recognition, and automated notifications to hospitals and relatives

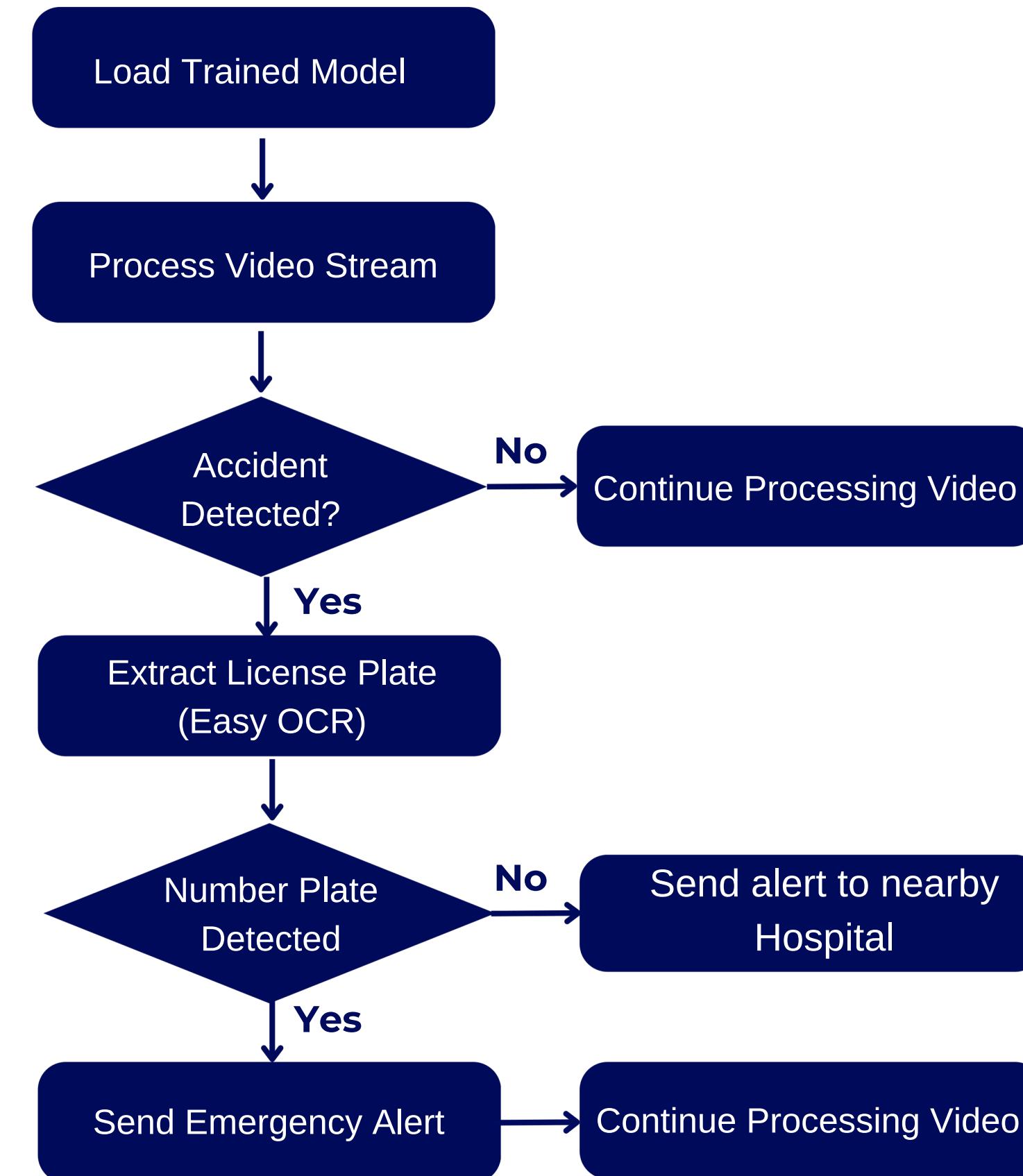
Literature Overview

- 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

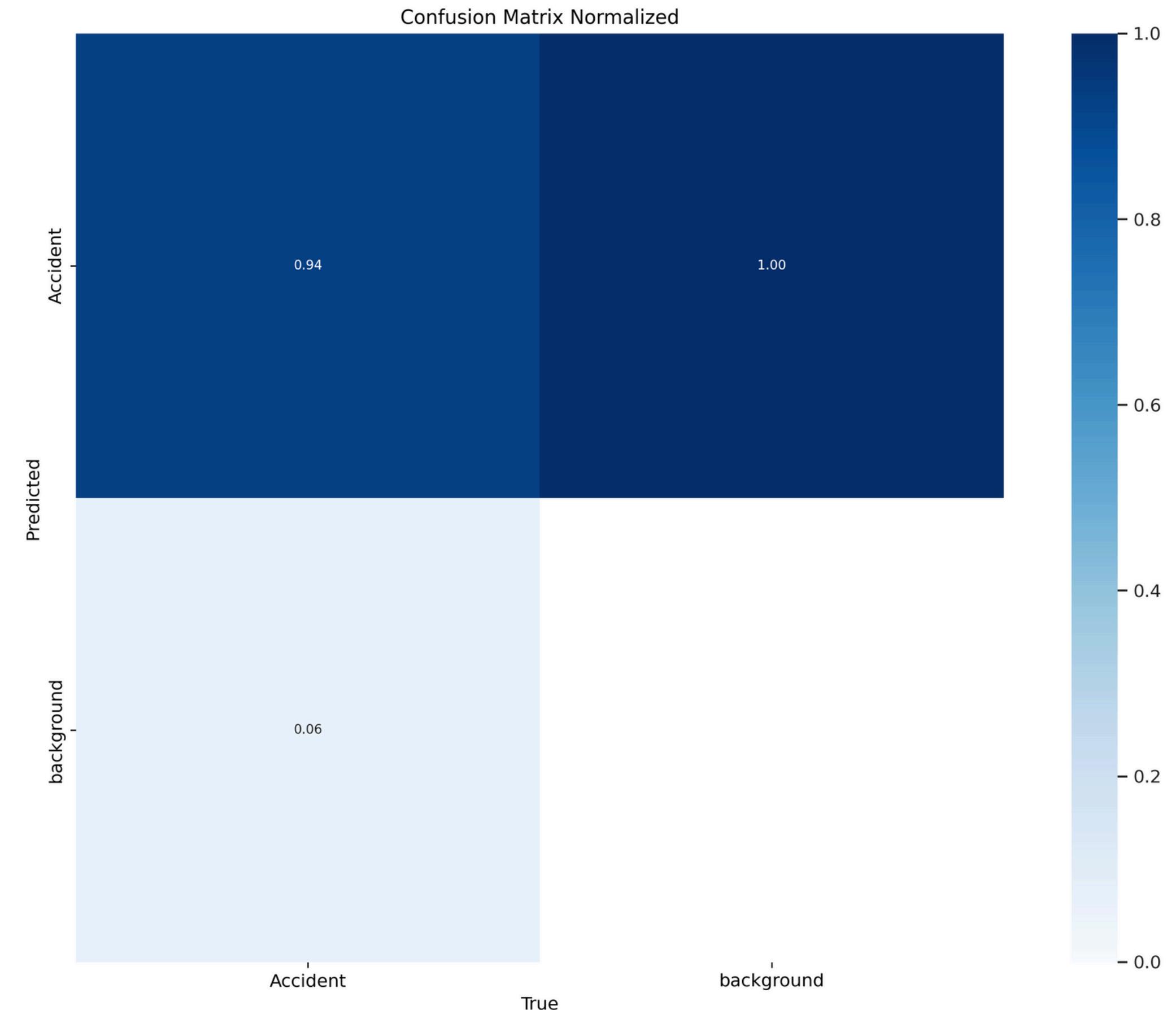
Model Evaluation

```
from ultralytics import YOLO

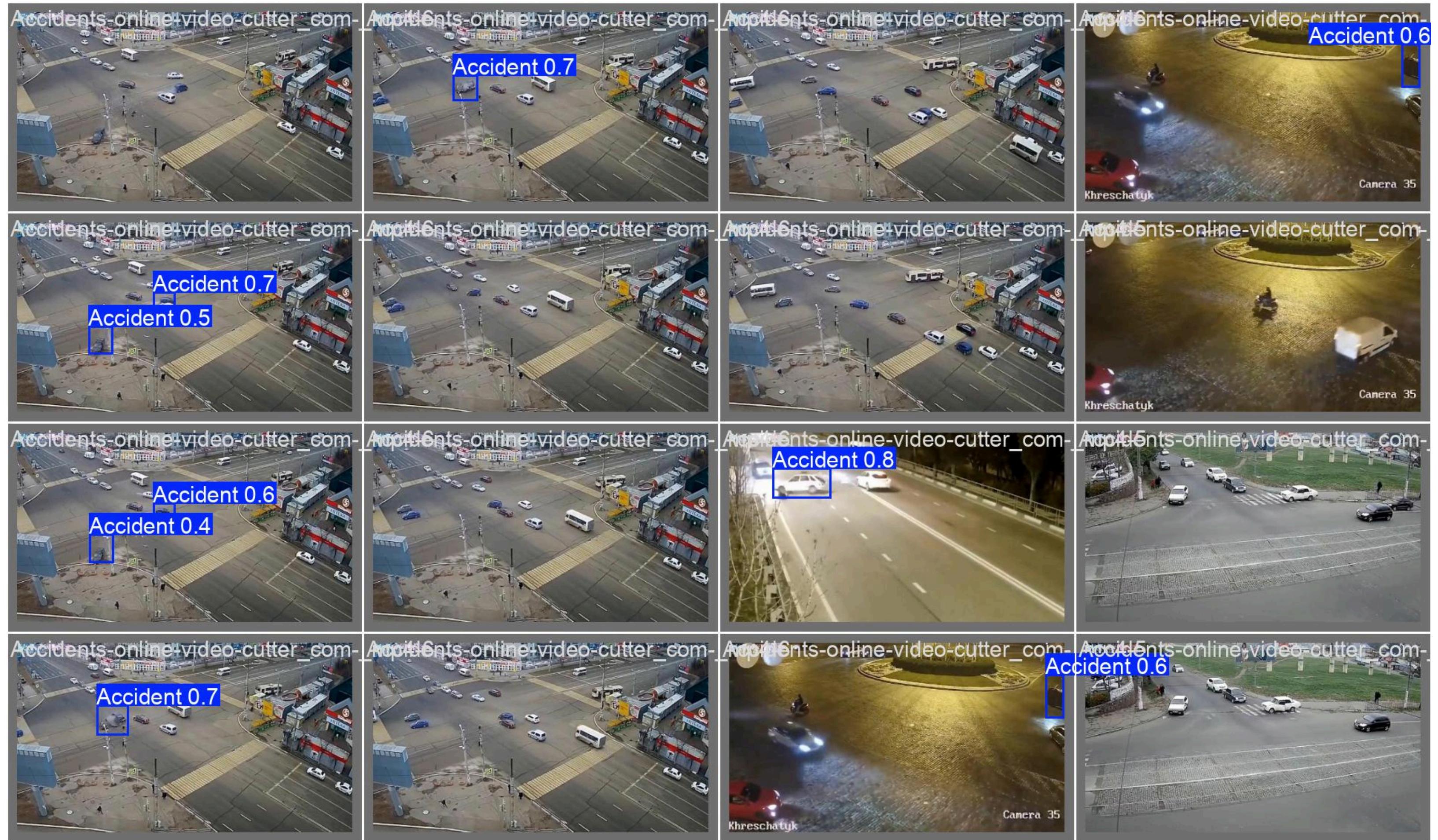
# Load your trained model
model = YOLO("/content/drive/MyDrive/Colab Notebooks/training_results/accidentmodel_22/weights/best.pt")

# 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
)

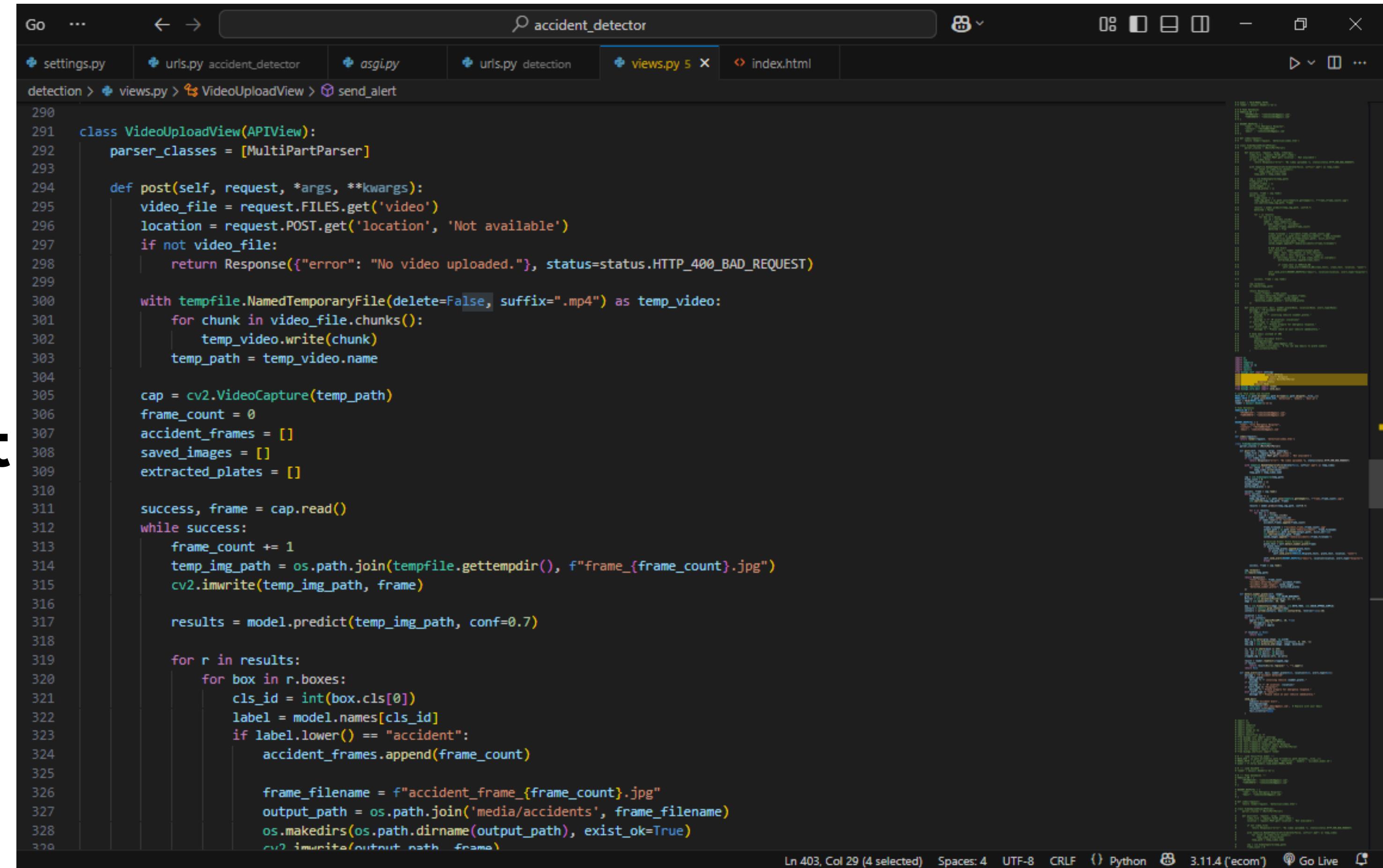
# After validation, 'metrics' will have all details
print(metrics)
```







Model Deployment

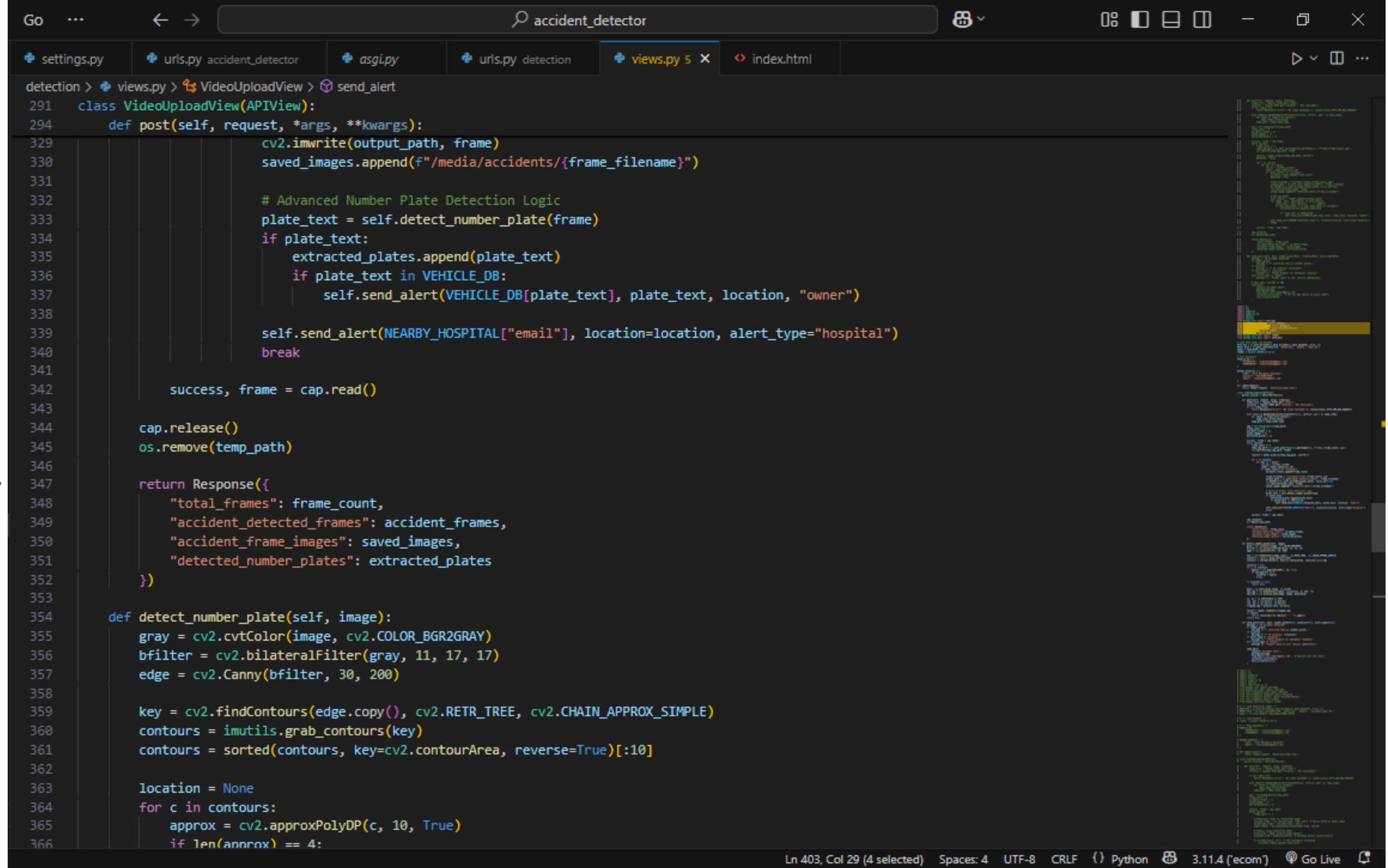


The screenshot shows a code editor window with the title bar "accident_detector". The search bar also contains "accident_detector". The tabs at the top include "settings.py", "urls.py accident_detector", "asgi.py", "urls.py detection", "views.py 5", and "index.html". The "views.py" tab is currently selected. The code in the editor is a Python class named "VideoUploadView" from the "views.py" file. The class handles a POST request for a video file. It uses a temporary file to store the uploaded video, reads frames from it using OpenCV, and performs predictions on each frame using a model. It identifies frames containing accidents and saves them to a media directory. The code uses various Python libraries like `tempfile`, `os`, `cv2`, and `model`.

```
290
291     class VideoUploadView(APIView):
292         parser_classes = [MultiPartParser]
293
294         def post(self, request, *args, **kwargs):
295             video_file = request.FILES.get('video')
296             location = request.POST.get('location', 'Not available')
297             if not video_file:
298                 return Response({"error": "No video uploaded."}, status=status.HTTP_400_BAD_REQUEST)
299
300             with tempfile.NamedTemporaryFile(delete=False, suffix=".mp4") as temp_video:
301                 for chunk in video_file.chunks():
302                     temp_video.write(chunk)
303             temp_path = temp_video.name
304
305             cap = cv2.VideoCapture(temp_path)
306             frame_count = 0
307             accident_frames = []
308             saved_images = []
309             extracted_plates = []
310
311             success, frame = cap.read()
312             while success:
313                 frame_count += 1
314                 temp_img_path = os.path.join(tempfile.gettempdir(), f"frame_{frame_count}.jpg")
315                 cv2.imwrite(temp_img_path, frame)
316
317                 results = model.predict(temp_img_path, conf=0.7)
318
319                 for r in results:
320                     for box in r.bboxes:
321                         cls_id = int(box.cls[0])
322                         label = model.names[cls_id]
323                         if label.lower() == "accident":
324                             accident_frames.append(frame_count)
325
326                             frame_filename = f"accident_frame_{frame_count}.jpg"
327                             output_path = os.path.join('media/accidents', frame_filename)
328                             os.makedirs(os.path.dirname(output_path), exist_ok=True)
329                             cv2.imwrite(output_path, frame)
```

Ln 403, Col 29 (4 selected) Spaces: 4 UTF-8 CRLF () Python 3.11.4 ('ecom') Go Live

Model Deployment



The screenshot shows a code editor window titled "accident_detector". The main pane displays Python code for a "VideoUploadView" class. The code handles video uploads, saves frames, detects number plates, and sends alerts. It uses OpenCV for image processing and a database for vehicle information. The code editor interface includes tabs for other files like "settings.py", "urls.py", "asgi.py", "urls.py", and "index.html". The bottom status bar shows file details: "Ln 403, Col 29 (4 selected) Spaces: 4 UTF-8 CRLF () Python 3.11.4 ('ecom') Go Live".

```
detection > views.py > VideoUploadView > send_alert
291     class VideoUploadView(APIView):
294         def post(self, request, *args, **kwargs):
299             output_path = os.path.join(settings.MEDIA_ROOT, 'temp')
300             temp_path = os.path.join(settings.MEDIA_ROOT, 'temp.jpg')
301             cap = cv2.VideoCapture(0)
302             frame_count = 0
303             accident_frames = []
304             saved_images = []
305             extracted_plates = []
306             VEHICLE_DB = {
307                 "KA01AA000": "Vehicle A",
308                 "KA01AA001": "Vehicle B",
309                 "KA01AA002": "Vehicle C"
310             }
311             NEARBY_HOSPITAL = {
312                 "email": "hospital@nearby.com"
313             }
314
315             while True:
316                 success, frame = cap.read()
317
318                 if not success:
319                     break
320
321                 frame_count += 1
322
323                 if frame_count % 10 == 0:
324                     accident_frames.append(frame)
325
326                 if frame_count % 50 == 0:
327                     saved_images.append(frame)
328
329                 cv2.imwrite(output_path, frame)
330                 saved_images.append(f"/media/accidents/{frame_filename}")
331
332                 # Advanced Number Plate Detection Logic
333                 plate_text = self.detect_number_plate(frame)
334                 if plate_text:
335                     extracted_plates.append(plate_text)
336                     if plate_text in VEHICLE_DB:
337                         self.send_alert(VEHICLE_DB[plate_text], plate_text, location, "owner")
338
339                     self.send_alert(NEARBY_HOSPITAL["email"], location=location, alert_type="hospital")
340
341             cap.release()
342             os.remove(temp_path)
343
344             return Response({
345                 "total_frames": frame_count,
346                 "accident_detected_frames": accident_frames,
347                 "accident_frame_images": saved_images,
348                 "detected_number_plates": extracted_plates
349             })
350
351             def detect_number_plate(self, image):
352                 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
353                 bfilter = cv2.bilateralFilter(gray, 11, 17, 17)
354                 edge = cv2.Canny(bfilter, 30, 200)
355
356                 key = cv2.findContours(edge.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
357                 contours = imutils.grab_contours(key)
358                 contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]
359
360                 location = None
361                 for c in contours:
362                     approx = cv2.approxPolyDP(c, 10, True)
363                     if len(approx) == 4:
```

Model Deployment

The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, ...
- Search Bar:** accident_detector
- Toolbar:** Includes icons for file operations like Open, Save, Find, and Run.
- Left Sidebar (Explorer):**
 - EXPLORER
 - ACCIDENT_DETECTOR
 - accident_detector
 - _pycache_
 - _init_.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
 - detection
 - _pycache_
 - migrations
 - models
 - templates\detection
 - index.html
 - _init_.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py (highlighted, 5 changes)
 - media
 - db.sqlite3
 - manage.py
- Central Area:** Code editor showing `views.py` content.

```
class VideoUploadView(APIView):
    def detect_number_plate(self, image):
        if location is None:
            return None

        mask = np.zeros(gray.shape, np.uint8)
        new_img = cv2.drawContours(mask, [location], 0, 255, -1)
        new_img = cv2.bitwise_and(image, image, mask=mask)

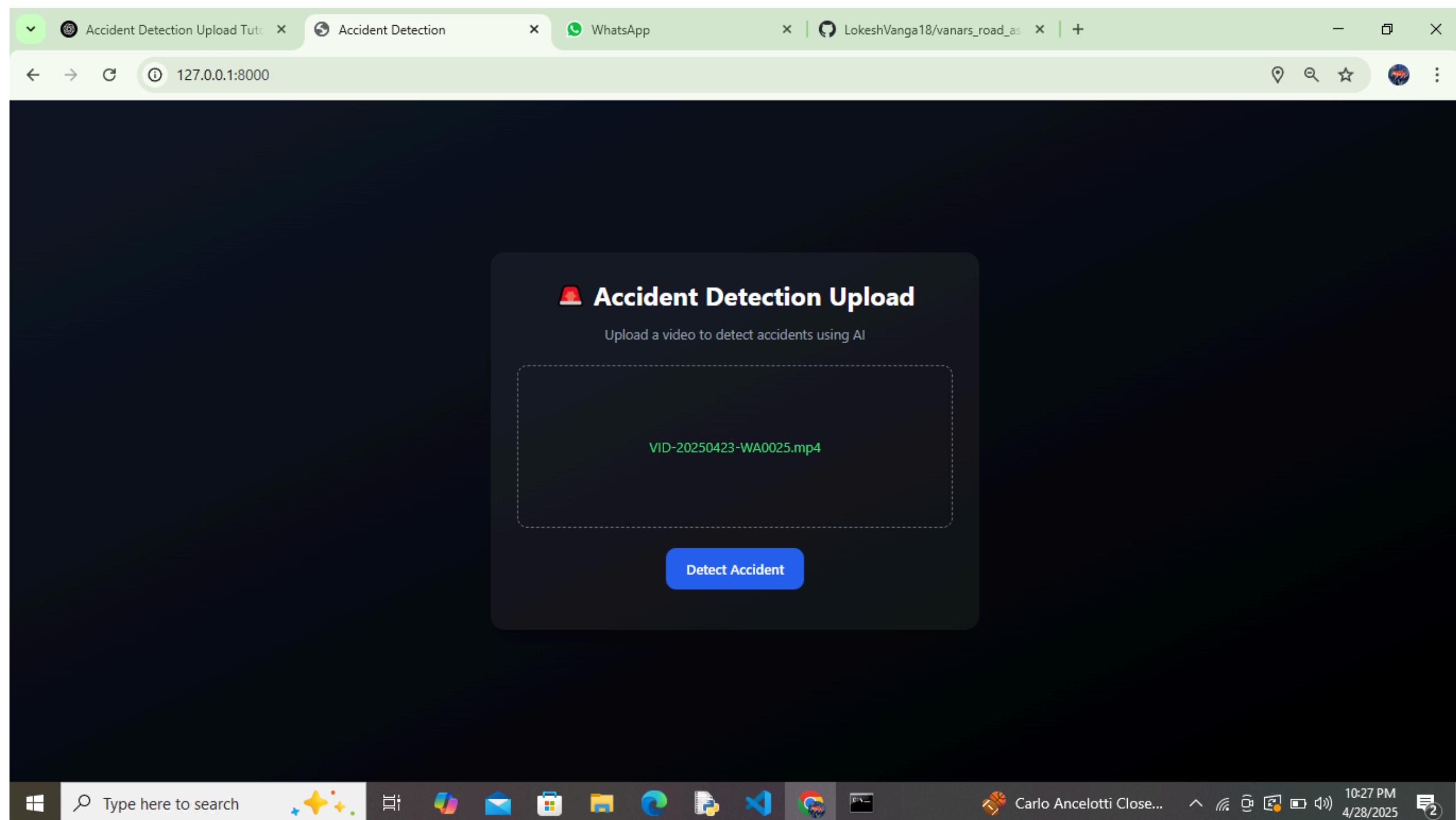
        (x, y) = np.where(mask == 255)
        (x1, y1) = (np.min(x), np.min(y))
        (x2, y2) = (np.max(x), np.max(y))
        cropped_img = gray[x1:x2+1, y1:y2+1]

        result = reader.readtext(cropped_img)
        if result:
            return result[0][-2].replace(" ", "").upper()
        return None

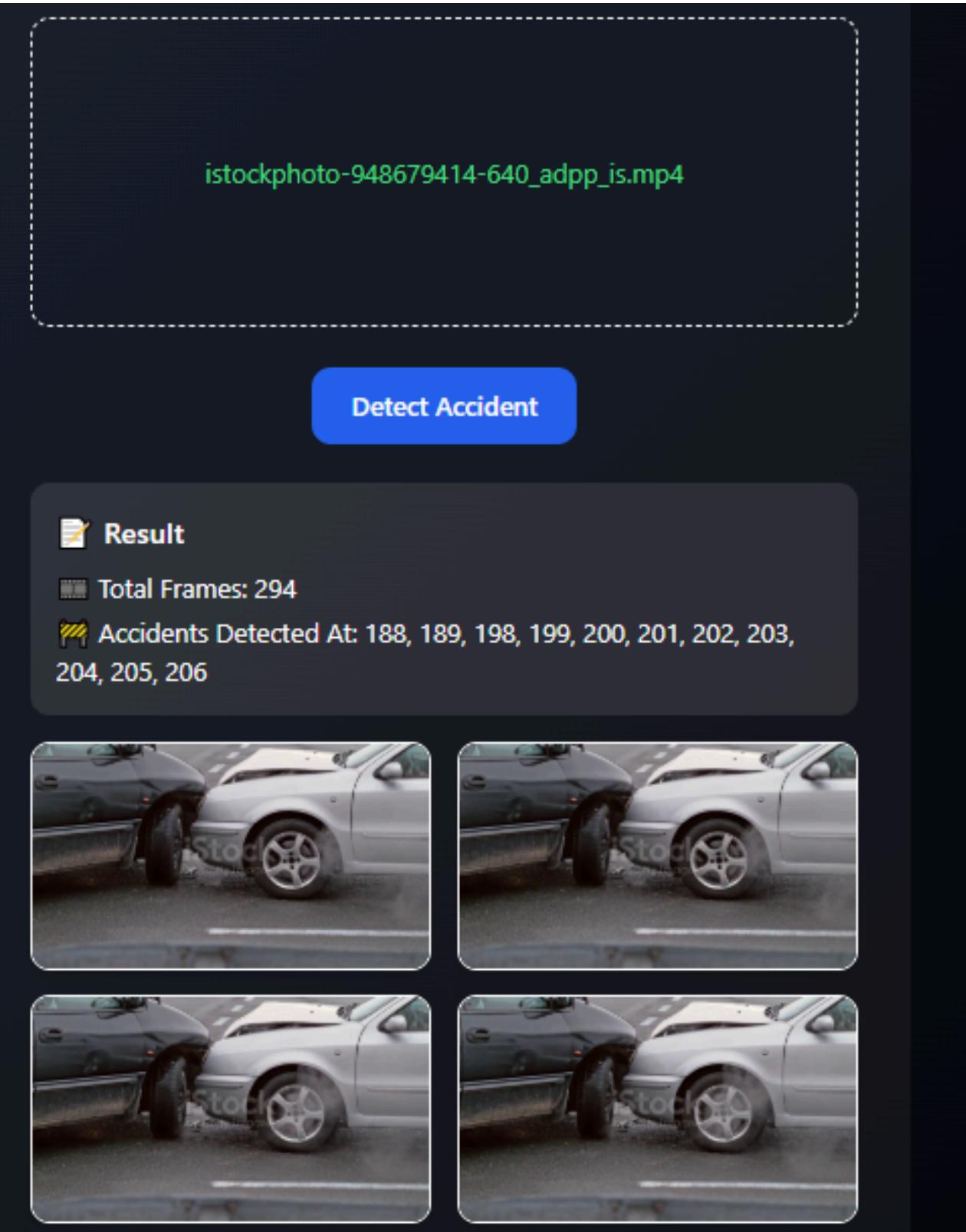
    def send_alert(self, mail, number_plate=None, location=None, alert_type=None):
        message = "⚠ Accident detected"
        if number_plate:
            message += f" involving vehicle {number_plate}."
        if location:
            message += f" ⚡ Location: {location}"
        if alert_type == "hospital":
            message += " Please prepare for emergency response."
        elif alert_type == "owner":
            message += " Please check on your vehicle immediately.

        send_mail(
            subject='Accident Alert',
            message=message,
            from_email='your_email@gmail.com', # Replace with your Gmail
            recipient_list=[mail],
            fail_silently=False,
        )
```
- Bottom Status Bar:** Ln 403, Col 29 (4 selected) Spaces: 4 UTF-8 CRLF Python 3.11.4 ('ecom') Go Live 10:18 PM 4/28/2025
- Taskbar:** Includes icons for Task View, Start, File Explorer, Task Manager, and a search bar.

Web Page Interface:



Initial results:



Accident Alert Inbox x

 n191052@rguktn.ac.in Apr 25, 2025, 2:35 PM (3 days ago)

to receiver_email ▾

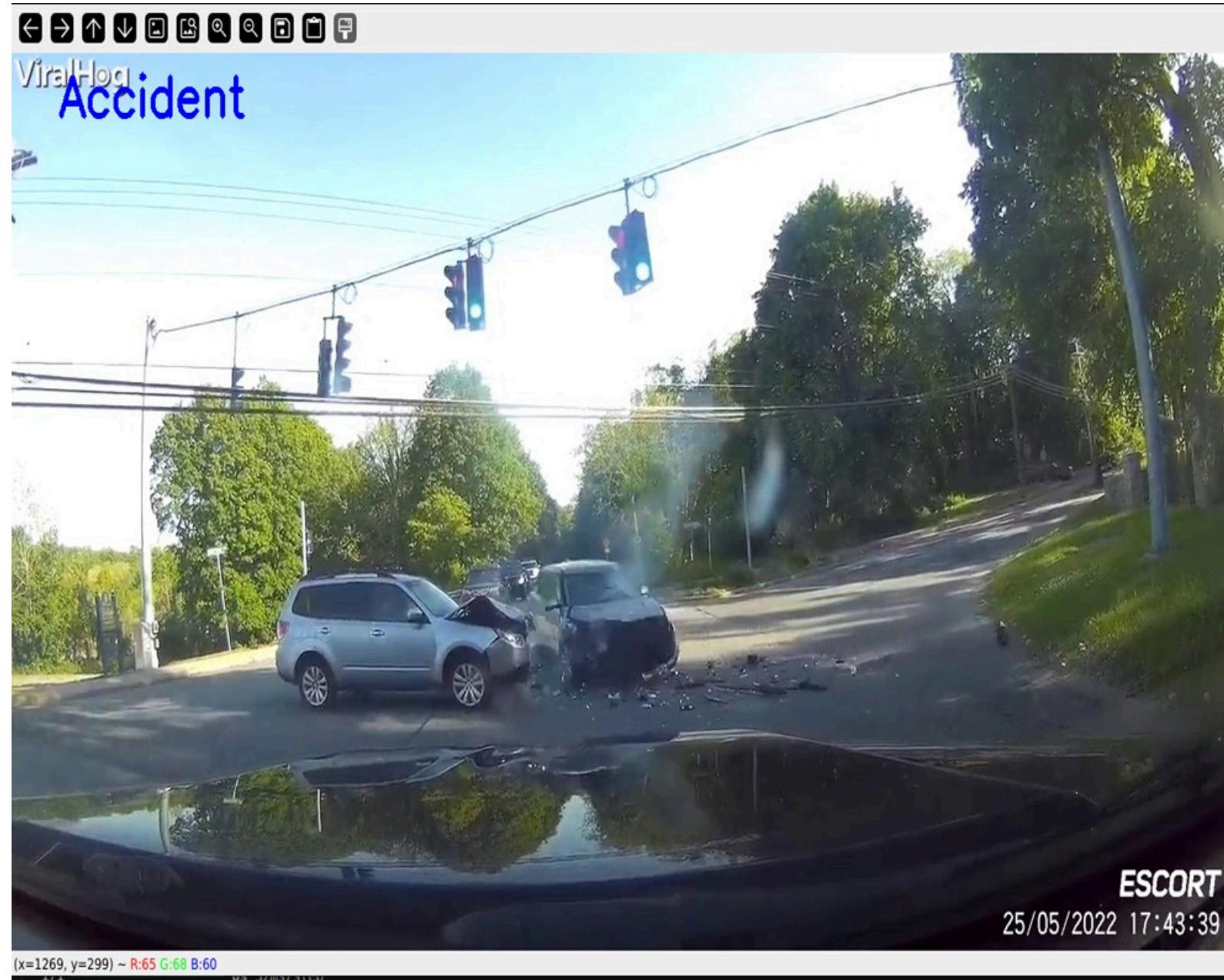
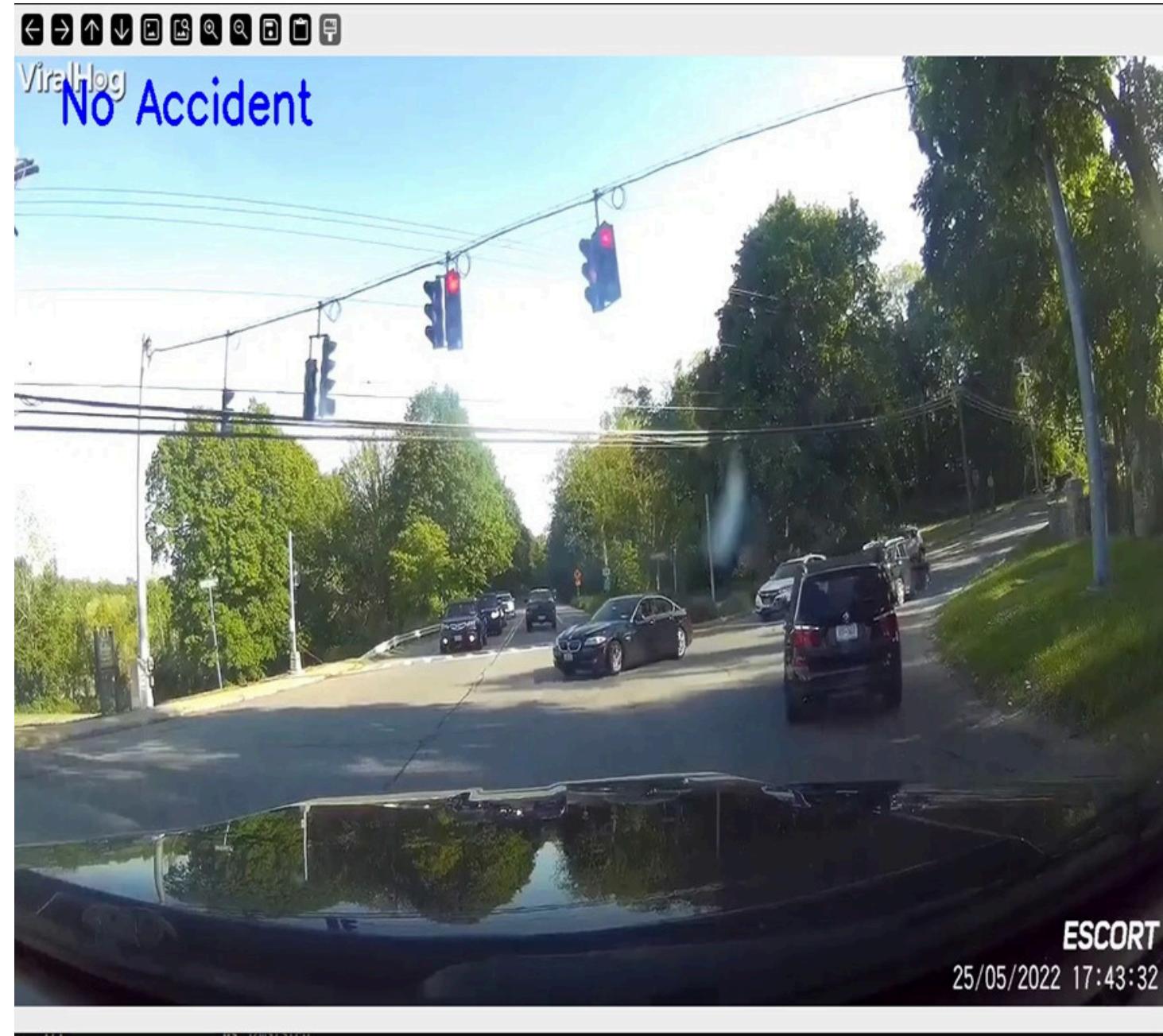
 Accident detected  Location: <https://www.google.com/maps?q=16.5061743,80.6480153> Please prepare for emergency response.



 n191052@rguktn.ac.in Apr 25, 2025, 2:36 PM (3 days ago)

to receiver_email ▾

...



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://arxiv.org/abs/2208.06461](https://arxiv.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!