Tutorial Report: Real-Time Traffic Congestion Detection using YOLOv5 and OpenCV

1. Introduction

This project implements a **real-time traffic congestion detection system** using YOLOv5 (a deep learning-based object detector) and OpenCV. The system processes webcam or video input, detects key objects (cars, buses, trucks, and people), and determines traffic congestion based on a configurable threshold.

---

2. System Overview

The pipeline includes:

1. **Model Loading** – Loads YOLOv5s from PyTorch Hub.
2. **Video Capture** – Processes live webcam feed or video file.
3. **Object Detection** – Detects vehicles and people in each frame.
4. **Congestion Detection** – Flags congestion when object count exceeds a threshold.
5. **Visualization** – Displays detection boxes and congestion status in real-time.
6. **Data Logging** – Saves logs with timestamps, object counts, and congestion status to CSV.

---

3. Requirements

3.1 Software & Libraries

- Python 3.x
- PyTorch (>=1.8)
- OpenCV (opencv-python)
- NumPy

3.2 Installation

pip install torch torchvision torchaudio opencv-python numpy

Note: YOLOv5 is loaded via torch.hub and automatically downloads the pretrained model.

---

4. Code & Explanations

4.1 Import Libraries

Explanation:

```
import cv2
import torch
import numpy as np
import time
import csv
from datetime import datetime
```

- torch loads the model.
- cv2 handles video processing.
- csv and datetime log detection outputs with timestamps.

4.2 Load YOLOv5 Model

```
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=
model.conf = 0.4  # confidence threshold
```

Explanation:

- Loads the lightweight YOLOv5s model.
- model.conf sets the minimum confidence level for detections.

4.3 Define Constants

```
CONGESTION_THRESHOLD = 10  # Adjust as needed
TRACK_CLASSES = [0, 2, 5, 7]  # person, car, bus, truck (COCO class
```

4.4 Setup CSV Logging

```
csv_file = open("congestion_data.csv", mode="w", newline="")
csv_writer = csv.writer(csv_file)
csv_writer.writerow(["Timestamp", "Object Count", "Congestion Statu
```

4.5 Process Video Input

```
video_path = 1  # Use 1 for webcam or replace with "video.mp4"
cap = cv2.VideoCapture(video_path)
```

4.6 Frame-by-Frame Detection Loop

```
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # YOLOv5 detection
    results = model(frame)
    detections = results.xyxy[0]  # [x1, y1, x2, y2, conf, class]

    # Count only tracked classes
    relevant_detections = [det for det in detections if int(det[5])
    object_count = len(relevant_detections)

    # Check congestion
    congestion = object_count > CONGESTION_THRESHOLD
```

```
    # Logging
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    csv_writer.writerow([timestamp, object_count, "Yes" if congesti

    # Draw detections
    for *xyxy, conf, cls in relevant_detections:
        label = f"{model.names[int(cls)]} {conf:.2f}"
        cv2.rectangle(frame, tuple(map(int, xyxy[:2])), tuple(map(i
        cv2.putText(frame, label, (int(xyxy[0]), int(xyxy[1]) - 5),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 2

    # Draw congestion status
    status_text = f"Congestion: {'YES' if congestion else 'NO'} | C
    cv2.putText(frame, status_text, (10, 30), cv2.FONT_HERSHEY_SIMP
                1, (0, 0, 255) if congestion else (0, 255, 0), 2)
```

---

5. Congestion Detection Logic

- The system watches for **COCO class IDs:**
    - 0: person

- 2: car
- 5: bus
- 7: truck

- **Threshold-based logic**:

```
congestion = object_count > CONGESTION_THRESHOLD
```

---

6. Sample Output

```
Timestamp: 2025-05-27 18:00:23 | Count: 12 | Congestion: YES
```

CSV (congestion_data.csv)

```
Timestamp,Object Count,Congestion Status
2025-05-27 18:00:23,12,Yes
2025-05-27 18:00:24,8,No
```

Visualization

- Bounding boxes with labels
- Real-time congestion status on screen

---

7. Output Files

File	Description
congestion_data.csv	Logs object count, congestion status, timestamp
Real-time display	Shows annotated video feed

---

8. Customization Tips

Task	How
Change congestion threshold	Edit CONGESTION_THRESHOLD
Use different object classes	Modify TRACK_CLASSES
Save processed video	Use cv2.VideoWriter
Run on GPU	Ensure torch uses CUDA (model.to('cuda'))

---

9. Potential Enhancements

- Add **ROI filtering** (detect congestion only in road areas)
- Integrate **alarm systems or SMS alerts**
- Build a **dashboard** using Flask or Streamlit
- Add **multi-camera** support
- Use **tracking algorithms** (e.g., DeepSORT) for better continuity

---

10. Conclusion

This YOLOv5 + OpenCV pipeline effectively detects real-time traffic congestion by counting vehicle-related objects in a video stream. It is lightweight, adaptable, and suitable for smart cities, traffic monitoring, and automated alert systems.

GitHub Repo (Example Placeholder)

GitHub - Real-Time-Traffic-Congestion-Detection