

**C.V RAMAN GLOBAL UNIVERSITY**

**BHUBANESWAR, ODISHA – 752054**

**CASE STUDY ON AIDS**

**TOPIC: TRAFFIQ-SMART TRAFFIC ANALYSIS**

|  |  |
| --- | --- |
| COURSE | B. Tech |
| BRANCH | CSE |
| SEMESTER | 6th |
| GROUP | 04 |
| SUB GROUP | 09 |
| UNDER SUPERVISION OF | Mr. Arib Nawal |



**TEAM MEMBERS**

|  |  |  |
| --- | --- | --- |
| **REG. NO.** | **NAME** | **CRANES REG.NO** |
| 2201020121 | SANIA HASSAN | **CL2025010601884931** |
| 2201020129 | SNEHA SINHA | **CL2025010601901472** |
| 2201020130 | SOFIA AKHTAR | **CL2025010601870311** |
| 2201020809 | SUBHASMITA SWAIN | **CL2025010601890688** |

# ACKNOWLEDGEMENT

We extend our heartfelt gratitude to all those who have contributed to the successful completion of this case study on **TraffiQ - Smart traffic analysis**. Their support and encouragement have been invaluable throughout this project.

First and foremost, we would like to express our sincere appreciation to our project supervisor, Mr. Arib Nawal, for his invaluable guidance and insightful feedback. His expertise and mentorship have been crucial in shaping our understanding of the subject and refining our approach to this case study.

We also acknowledge the collaborative efforts of our team members, whose dedication and commitment to this project have significantly enriched its development and outcome. Their collective enthusiasm and hard work were instrumental in overcoming challenges and achieving our objectives.

Finally, we extend our thanks to the faculty and administration of C. V. Raman Global University for providing us with the resources and an environment that fosters academic growth and innovation. This case study would not have been possible without the support and opportunities offered by our university.

# TABLE OF CONTENT

* + 1. ABSTRACT 5
    2. INTRODUCTION 6
    3. PROBLEM STATEMENT 6 4. TOOLS & TECHNOLOGIES USED 6

1. DATASET OVERVIEW 7
2. METHODOLOGY 8-9
3. SOURCE CODE 10-12
4. OUTPUT 13
5. REAL LIFE APPLICATIONS 14-16
6. CONCLUSION 17
7. REFERENCES 17

# ABSTRACT

This case study presents that TraffiQ is an AI-powered smart traffic analysis system that uses computer vision technology to transform urban traffic monitoring. The system automatically detects safety violations like missing helmets and seatbelts, measures vehicle speeds, and classifies vehicles by type (two-wheelers, cars, and heavy vehicles). All this information is presented through an intuitive dashboard that provides real-time insights for traffic management and law enforcement.

Built using Python and modern AI frameworks like YOLO, TraffiQ addresses the limitations of manual traffic surveillance by creating a scalable, automated solution. The system processes video feeds from traffic cameras, applies machine learning models to identify objects and behaviors of interest, and generates actionable data that can help reduce accidents and improve traffic flow. Its real-time processing capabilities make it particularly valuable for busy urban environments where traditional monitoring methods struggle to keep pace.



**INTRODUCTION**

As cities grow busier with more vehicles, old ways of managing traffic aren't working well anymore. TraffiQ offers a smart solution that watches traffic automatically, helping keep roads safer and less congested.



**PROBLEM STATEMENT:**

Manual traffic monitoring has several problems:

\* Can't catch safety violations (missing helmets/seatbelts) consistently

\* No good way to measure vehicle speeds in real-time

\* Difficult to count and categorize different vehicle types

\* Lacks a central system to show useful traffic patterns

# TOOLS AND TECHNOLOGIES USED:

Tool used for programming : Google Collab , Hugging Face , Streamlit

Programming Language: Python

Libraries:

- OpenCV (for processing video)

- TensorFlow/PyTorch (for AI models)

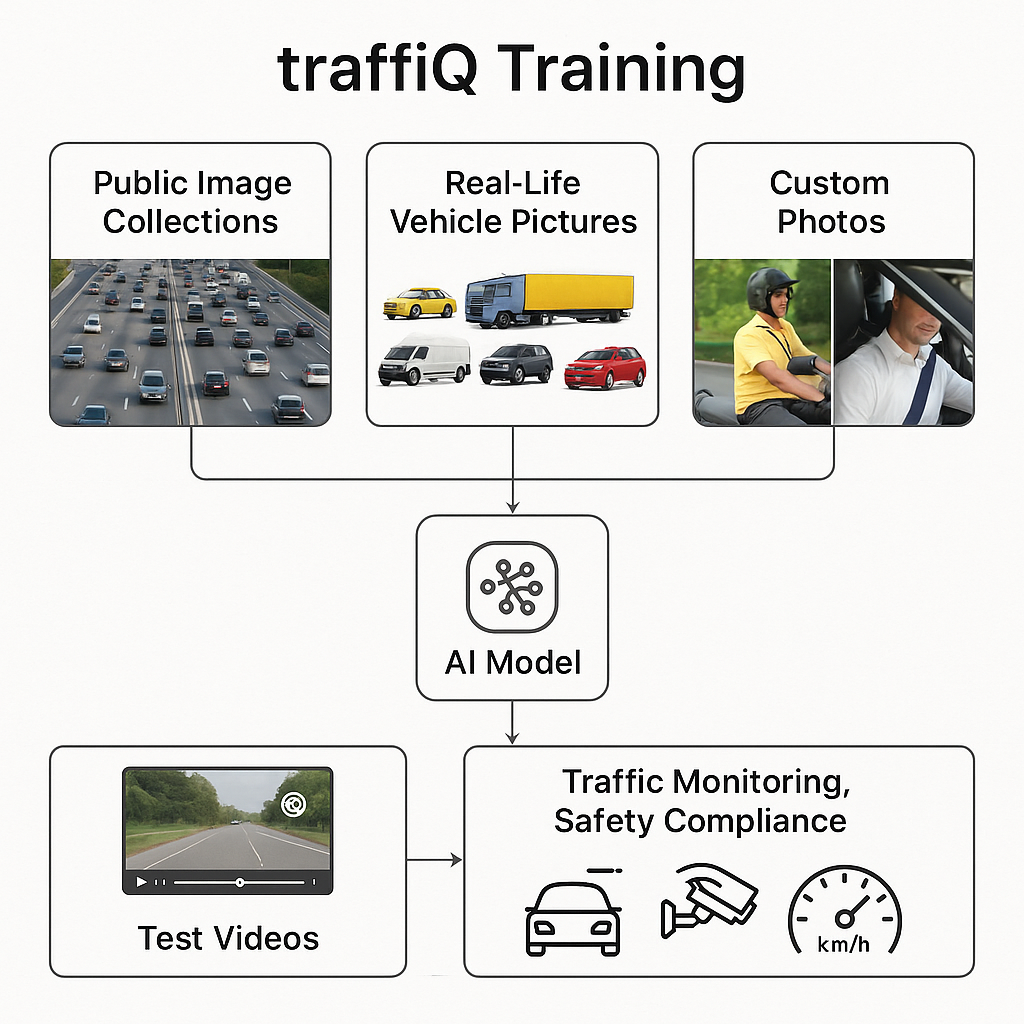
- YOLO (for detecting objects)

- Streamlit (for creating dashboards)



**DATASET OVERVIEW:**

Our traffiQ smart traffic detection system was developed using a comprehensive training dataset to ensure exceptional accuracy and reliability in diverse traffic scenarios. We built our AI model on a foundation of public image collections, supplemented with real-life vehicle pictures to maximize recognition capabilities across different vehicle types and conditions. To specifically address safety compliance monitoring in Indian traffic contexts, we incorporated custom photos highlighting helmet and seatbelt usage patterns.



Additionally, specialized test videos were utilized to calibrate and validate our speed measurement functionality. This meticulously curated training approach enables traffiQ to deliver precise real-time traffic monitoring, safety enforcement, and analytics in even the most challenging urban environments.

# METHODOLOGY/ALGORITHM USED:

**Step 1: Data Loading**

**1. Load Input Video Streams:**

* Real-time or pre-recorded traffic footage is loaded into the system.
* Each video is segmented into frames for analysis.
* Example sources include roadside cameras or drone footage.

**Step 2: Data Preprocessing**

**2. Frame Preprocessing and Feature Extraction:**

* Each frame is processed to enhance visibility and reduce noise.
* Object detection algorithms (e.g., YOLO, Faster R-CNN) are used to identify:
  + Vehicles
  + Riders/Drivers
  + Helmets and seatbelts
* Extracted features include position, size, and detected object class.

**Step 3: Safety Violation Detection**

**3. Detect Helmet and Seatbelt Usage:**

* Analyze extracted features to check for:
  + Riders without helmets
  + Drivers without seatbelts
* Violations are marked using colored bounding boxes (e.g., red for no helmet).
* Violations are logged with frame ID and timestamp.

**Step 4: Speed Tracking**

**4. Vehicle Speed Estimation:**

* Track vehicle position across multiple frames using object tracking.
* Calculate displacement over time to estimate speed.
* Speed = Distance Traveled / Time Elapsed
* Speed is compared against legal limits for compliance.

**Step 5: Vehicle Classification**

**5. Classify Vehicle Types:**

* Use pre-trained classification models to categorize:
  + Two-wheelers (bikes)
  + Four-wheelers (cars)
  + Heavy vehicles (trucks, buses)
* Assign labels and use these for analytics and filtering.

**Step 6: Result Aggregation**

**6. Aggregate and Update Dashboard Data:**

1. Compile data on:
   * Violation count per frame
   * Speed of each vehicle
   * Types and count of vehicles
2. Feed data to a centralized dashboard.

**Step 7: Visualization & Display**

**7. Display Results in Dashboard:**

* Real-time dashboard shows:
  + Live video feed with overlays (speed, vehicle type, violation box)
  + Count summaries (e.g., “12 bikes, 2 violations”)
  + Graphs for traffic density, average speed, violation trends

# SOURCE CODE:

import cv2

import math

from ultralytics import YOLO

# Constants

FPS = 30 # Use actual FPS of the video if different

PIXELS\_PER\_METER = 10 # Adjust this based on your video

model = YOLO("yolov8n.pt")

# Load video

video\_path = "clip6.mp4"

cap = cv2.VideoCapture(video\_path)

# Output video writer

width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

fps = int(cap.get(cv2.CAP\_PROP\_FPS)) or FPS

fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')

out = cv2.VideoWriter("output\_video6.mp4", fourcc, fps, (width, height))

# Vehicle tracking dictionary {id: (prev\_center, speed)}

track\_data = {}

# Vehicle classes of interest

vehicle\_classes = ["car", "motorcycle", "bus", "truck", "auto"]

vehicle\_colors = {

"car": (255, 0, 255),

"motorcycle": (255, 0, 0),

"bus": (0, 0, 255),

"truck": (0, 125, 255),

#"auto": (255, 125, 0)

}

def draw\_label\_with\_background(img, text, topleft, color, font\_scale=1.0, thickness=2):

font = cv2.FONT\_HERSHEY\_SIMPLEX

text\_size, \_ = cv2.getTextSize(text, font, font\_scale, thickness)

text\_w, text\_h = text\_size

x, y = topleft

# Draw filled rectangle (label background)

cv2.rectangle(img, (x, y - text\_h - 10), (x + text\_w+5, y), color, -1)

# Put text over it

cv2.putText(img, text, (x, y - 5), font, font\_scale, (255, 255, 255), thickness, lineType=cv2.LINE\_AA)

def estimate\_speed(p1, p2):

dist = math.hypot(p2[0] - p1[0], p2[1] - p1[1]) # Euclidean distance

meters = dist / PIXELS\_PER\_METER

speed = meters \* fps \* 3.6 # km/h

return round(speed, 2)

frame\_count = 0

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

frame\_count += 1

# Detect objects

results = model(frame, conf=0.5, verbose=False)[0]

for i, box in enumerate(results.boxes):

cls\_id = int(box.cls[0])

class\_name = model.names[cls\_id]

xyxy = box.xyxy[0].cpu().numpy().astype(int)

x1, y1, x2, y2 = xyxy

center = ((x1 + x2) // 2, (y1 + y2) // 2)

if class\_name in vehicle\_classes:

color = vehicle\_colors.get(class\_name, (255, 255, 255))

# Estimate speed using center point tracking

obj\_id = f"{class\_name}\_{i}" # basic ID for simplicity

prev\_center = track\_data.get(obj\_id, (center, 0))[0]

speed = estimate\_speed(prev\_center, center) if frame\_count > 1 else 0

track\_data[obj\_id] = (center, speed)

# Draw rectangle and label

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

label = f"{class\_name} {speed} km/h"

draw\_label\_with\_background(frame, label, (x1, y1), color)

elif class\_name == "person":

person\_box = box.xyxy[0]

has\_helmet = False

# Check for helmets in the same frame

for other in results.boxes:

other\_cls = int(other.cls[0])

other\_name = model.names[other\_cls]

if other\_name == "helmet":

helmet\_box = other.xyxy[0]

iou = (

max(0, min(person\_box[2], helmet\_box[2]) - max(person\_box[0], helmet\_box[0])) \*

max(0, min(person\_box[3], helmet\_box[3]) - max(person\_box[1], helmet\_box[1]))

)

if iou > 0:

has\_helmet = True

break

if not has\_helmet:

draw\_label\_with\_background(frame, "no Helmet", (int(person\_box[0]), int(person\_box[1])), (0, 0, 255))

cv2.rectangle(frame, (int(person\_box[0]), int(person\_box[1])),

(int(person\_box[2]), int(person\_box[3])), (0, 0, 255), 2)

out.write(frame)

# Optional: show live frame while processing

# cv2.imshow("Traffic Detection", frame)

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

# break

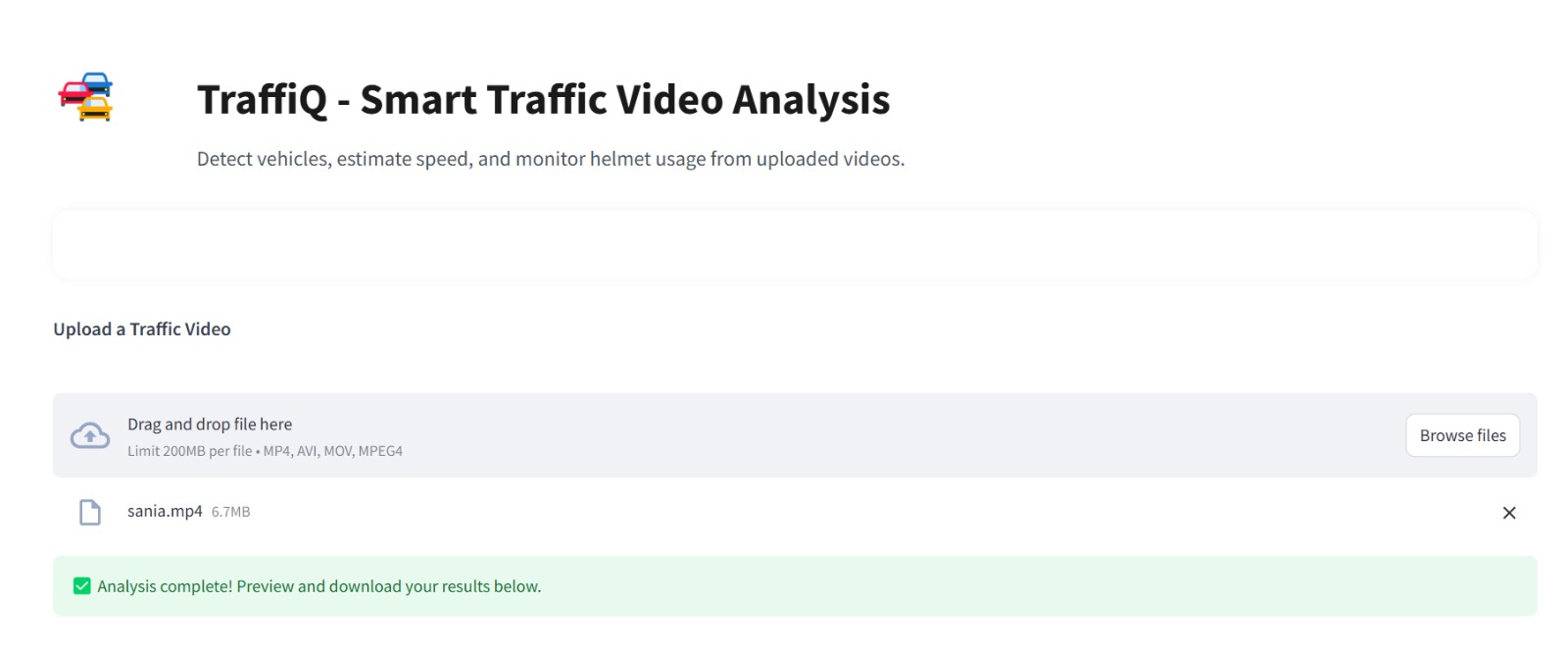
cap.release()

out.release()

cv2.destroyAllWindows()

print("✅ Done! Output saved as output\_video3.mp4")

# OUTPUT



# REAL LIFE APPLICATIONS OF K-MEANS CLUSTERING

# Integration with Traffic Lights

# Real-time traffic signal optimization based on current road conditions

# Adaptive timing systems that respond to changing traffic volumes

# Priority corridors for emergency vehicles through automated signal adjustments

# Pedestrian-responsive crosswalk management based on actual foot traffic

# Coordinated signal timing across multiple intersections to create "green waves"

# Dynamic lane allocation during peak hours

# Event-specific traffic management plans for stadiums, concerts, and public gatherings

# Monitoring Dangerous Intersections

# Real-time incident detection with automated alert systems

# Historical pattern analysis to identify time-specific risk factors

# Visibility and blind spot assessment at problematic junctions

# Near-miss analytics to address issues before accidents occur

# Weather-related risk amplification monitoring

# Vehicle-pedestrian interaction tracking in high-conflict zones

# Behavioral analysis of drivers at specific intersections

# Effectiveness measurements for implemented safety improvements

# Support for Law Enforcement

# Automated detection of speed limit violations with evidence capture

# Red light running identification and documentation

# Helmet and seatbelt compliance monitoring

# Dangerous driving behavior pattern recognition

# License plate recognition for stolen vehicles or wanted individuals

# HOV/carpool lane compliance verification

# Illegal parking and stopping detection

# Evidence preservation for traffic violation adjudication

# Distracted driving identification capabilities

# Improved Road Planning

# Vehicle classification and volume analytics by time and location

# Origin-destination mapping to understand travel patterns

# Peak demand analysis for capacity planning

# Modal split tracking (cars, trucks, buses, motorcycles)

# Road utilization efficiency assessments

# Wear pattern analysis for maintenance prioritization

# Impact studies of construction or closures on surrounding roads

# Future growth projection modeling based on historical trends

# Seasonal and event-based traffic flow variations

# Smart City Policy Development

# Data-driven congestion pricing models

# Public transportation routing optimization based on demand patterns

# Environmental impact assessments of traffic patterns

# Alternative transportation infrastructure planning support

# Economic impact analysis of traffic conditions on local businesses

# Accessibility measurements for different city regions

# Equity analysis of transportation resources across neighborhoods

# Benchmarking against other similar urban areas

# Policy effectiveness measurement through before/after comparisons

# Scenario modeling for proposed traffic management strategies

# Public health correlation studies with traffic patterns and pollution

# CONCLUSION

TraffiQ provides a comprehensive solution for monitoring and analyzing traffic using advanced AI technology. By enhancing road safety and improving traffic flow, it addresses key challenges faced in urban mobility. Its ability to detect violations, track vehicle speeds, and categorize vehicle types in real-time makes it an essential tool for modern traffic management. As TraffiQ continues to evolve, it holds the potential to become a vital component in the development of smart cities, contributing to safer, more efficient, and data-driven urban environments.

# REFERENCE

*  **YOLO Documentation**  
  Explore the official Ultralytics YOLO documentation for real-time object detection and image segmentation:  
  🔗 [Ultralytics YOLO Docs](https://docs.ultralytics.com/)​[Home](https://docs.ultralytics.com/?utm_source=chatgpt.com)
*  **OpenCV Website**  
  Access the Open Source Computer Vision Library for real-time computer vision applications:  
  🔗 [OpenCV Official Site](https://opencv.org/)​[OpenCV+3SourceForge+3GitHub+3](https://sourceforge.net/projects/opencvlibrary/?utm_source=chatgpt.com)
*  **Streamlit Resources**  
  Build and deploy interactive data apps with Streamlit's open-source Python framework:  
  🔗 [Streamlit Documentation](https://docs.streamlit.io/)​[Learn R, Python & Data Science Online+2Streamlit+2Streamlit Docs+2](https://streamlit.io/?utm_source=chatgpt.com)
*  **Helmet Detection Dataset**  
  Utilize the YOLOv8 Safety Helmet Detection dataset available on Kaggle:  
  🔗 [YOLOv8 Safety Helmet Detection](https://www.kaggle.com/code/plasticglass/yolov8-safety-helmet-detection)​
*  **Traffic Analysis Research Papers**  
  Review comprehensive studies on network traffic analysis and prediction techniques:  
  🔗 [A Review of Network Traffic Analysis and Prediction Techniques](https://www.researchgate.net/publication/339927785_A_review_of_network_traffic_analysis_and_prediction_techniques)