**Source Code -Intelligent Traffic Management:**

**Python**:

* Widely used for developing machine learning models and data analysis due to its extensive libraries (e.g., TensorFlow, Keras, Scikit-learn).
* Ideal for handling real-time data processing and implementing AI algorithms where low-level hardware interaction is required.

**Code:**

import os

import cv2

import torch

# Load the custom-trained YOLO model

model=torch.hub.load('ultralytics/yolov5','custom', path='C:/Users/Rajanna/AppData/Local/Programs/Python/Python312/yolov5/runs/train/exp9/weights/best.pt')

# Define the path to the folder containing your images image\_folder = 'C:/Users/Rajanna/AppData/Local/Programs/Python/Python312/dataset/dataset/images' # Replace with your actual folder path

# Get all image paths in the folder

image\_paths = [os.path.join(image\_folder, filename) for filename in os.listdir(image\_folder) if filename.endswith(('.jpg', '.jpeg', '.png'))]

if not image\_paths:

print("No images found in the specified folder.")

else:

print(f"Found {len(image\_paths)} images to process.")

class\_names = model.names # Custom class names inferred from your model

road\_info = []

for i, image\_path in enumerate(image\_paths):

image = cv2.imread(image\_path)

if image is None:

print(f"Error loading image from {image\_path}")

continue

image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

results = model(image\_rgb)

detected\_objects = results.xyxy[0].numpy()

vehicle\_count = len(detected\_objects)

congestion = (vehicle\_count / 100) \* 100 # Adjusted for max capacity

congestion\_level = "High" if congestion > 30 else "Moderate" if congestion > 10 else "Low"

# Check for emergency vehicles

emergency\_vehicle\_detected = any(

class\_names[int(obj[5])] == 'emergency' for obj in detected\_objects

)

if emergency\_vehicle\_detected:

time\_for\_emergency = 100 / 20 # Update with your specific values