In [ ]:
```
!pip install ultralytics -q # Importing necessary libraries import cv2
from ultralytics import YOLO
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import os import subprocess
from tqdm.notebook import tqdm
from IPython.display import Video, display
# Define a function to resize the frame def resize_frame(frame,
scale_percent):
 width = int(frame.shape[1] * scale_percent / 100) height =
int(frame.shape[0] * scale_percent / 100) dim = (width, height)
 resized = cv2.resize(frame, dim, interpolation=cv2.INTER_AREA)
return resized
# Load the YOLO model
model = YOLO('yolov8s.pt')
```

In [ ]:
```
# Load class names
class_names = model.model.names
# Define the video path
video_path = ‘content/video1.mp4’
# Display the target video frac = 0.65
display(Video(data=video_path, height=int(720 * frac), width=int(1280 * frac
# Define configurations verbose = False scale_percent = 50 cy_linha = int(15
cx_sentido = int(2000 * scale_percent / 100) offset = int(8 * scale_percent
contador_in = 0 contador_out = 0 veiculos_contador_in = {2: 0, 3: 0, 5: 0, 7
{2: 0, 3: 0, 5: 0, 7: 0} frames_list = []
# Open video capture
video_capture = cv2.VideoCapture(video_path)
# Define video writer
output_video_path = 'result.mp4'
output_video = cv2.VideoWriter(output_video_path, cv2.VideoWriter_fourcc(*'m
#Loop through video frames
for _ in tqdm(range(int(video_capture.get(cv2.CAP_PROP_FRAME_COUNT)))):
 _, frame = video_capture.read()
 frame = resize_frame(frame, scale_percent)
 # Predict using YOLO model results = model(frame)
 # Get bounding boxes, confidences, and classes boxes = results.xyxy[0].cpu(
= results.xyxy[0, :, 4] classes = results.xyxy[0, :, 5]
 # Iterate through detections
```

In [ ]:
```
for box, confidence, class_id in zip(boxes, confidences, classes):
 # Extract box coordinates
 xmin, ymin, xmax, ymax = box.astype(int)
 # Calculate center of bounding box center_x = (xmin + xmax) // 2 center_y =
// 2
 # Draw bounding box and label
 cv2.rectangle(frame, (xmin, ymin), (xmax, ymax), (0, 255, 0), 2)
 cv2.putText(frame, class_names[int(class_id)], (xmin, ymin - 10), cv2.FONT_
0.5, (0, 255, 0), 2)
 # Check if center is within transition line if cy_linha - offset < center_y
if 0 < center_x < cx_sentido: contador_in += 1 veiculos_contador_in[int(clas
else:
 contador_out += 1
 veiculos_contador_out[int(class_id)] += 1
 # Update frame with counting information
 cv2.putText(frame, f'In: {contador_in}', (30, cy_linha + 60), cv2.FONT_HERS
255, 0), 2)
 cv2.putText(frame, f'Out: {Contador _ out}', (30, cy_linha - 40), cv2.FONT_
(255, 255, 0), 2)
 # Save frame to list frames_list.append(frame)
 # Write frame to output video output_video.write(frame)
# Release video capture and writer video_capture.release() output_video.rele
# Show sample processed frames for i in [28, 29, 32, 40, 42, 50, 58]:
 plt.figure(figsize=(14, 10)) plt.imshow(frames_list[i])
 plt.show()
Display output video display(Video(data=output_video_path, embed=True)
```

In [ ]:

In [ ]: