University of East London

Report: Intelligent Traffic Management System

Module: Data Science & AI (Level 5)

Prepared For: Nitish Chooramun

Question No: 2

Prepared by: **Suraj Kumar Shah (2848224) Group: 4**

PROJECT DEPLOYMENT

- **Primary System:** Live | Github
- **Extended Prototype:** Live | Github

Contents

1. Executive Summary 3
2. Phase 1: Business Understanding (Problem Definition) 3
   2.1 Objective 3
   2.2 Problem Statement 3
   2.3 Proposed Solution 3
3. Phase 2: Data Mining (Data Sources) 3
   3.1 Primary Data Sources 3
   3.2 Secondary Data Sources 3
   3.3 Dataset Used 4
4. Phase 3: Data Cleaning and Preparation 4
   4.1 Data Filtering and Normalization 4
   4.2 Coordinate Conversion Logic (Code Snippet) 4
   4.3 Data Distribution Analysis 5
5. Phase 4: Modeling (Techniques for Detection) 5
   5.1 Technique Justification 5
   5.2 Training Configuration and Hyperparameters 5
   5.3 Model Training Implementation (Code Snippet) 6
   5.4 Evaluation Results 6
6. Phase 5: Deployment (Counting Techniques) 6
   6.1 Inference and Counting Logic 6
   6.2 Detection Implementation (Code Snippet) 7
7. Phase 6: Reporting and Visualization 8
   7.1 Automated Dashboarding 8
   7.2 Report Generation Implementation (Code Snippet) 8
8. Conclusion 9
9. References 9

1. Executive Summary

A very crucial infrastructure in the world is transportation. With the growth of cities, traffic and congestion management is an issue of concern. This report defines the technical proposal that Apex Research has come up with to deploy an Intelligent Traffic Management System.

2. Phase 1: Business Understanding (Problem Definition)

2.1 Objective

The primary goal is to transition from manual or sensor-based traffic counting to a Vision-based AI solution. This will enable automated, continuous monitoring of traffic flow.

2.2 Problem Statement

Law enforcement does not have fine-grained information regarding the vehicle composition (e.g., the percentage of heavy trucks and cars) at a particular intersection, flyover, and crossroad. In the absence of this data, it is inefficient to optimize the timelines of traffic lights or to expand roads.

2.3 Proposed Solution

Certain Deep learning models will be applied to process video feeds, which will recognize and classify vehicles in real-time to produce traffic density reports.

3. Phase 2: Data Mining (Data Sources)

To develop a strong model, we need different sources of data that would represent traffic in different angles and conditions.

3.1 Primary Data Sources

- **CCTV Traffic Cameras:** There are cameras at the crossroads that are stationary and offer lateral views of traffic.
- **UAV/Drone Imagery:** Air camera shots to examine complexity of roads such as a roundabout and flyover where a stationary camera does not see.

3.2 Secondary Data Sources

- **Inductive Loop Sensors:** Physical sensors embedded in roads to validate the counts generated by our vision model.

3.3 Dataset Used

For this pilot project, we utilized the **VisDrone2019-DET** dataset. This dataset is specifically designed for drone-based traffic analysis, offering a high degree of difficulty due to small objects, occlusions, and varying weather conditions.

---

4. Phase 3: Data Cleaning and Preparation

4.1 Data Filtering and Normalization

Raw camera data are typically not in a format that can be used by a typical AI model. The VisDrone dataset is annotated in a special format x y width height which cannot be easily used in modern object detectors and does not have normalized center-coords (as needed by YOLO).

Key Preparation Steps:

1. **Filtering:** We narrowed down on 10 classes namely: pedestrian, people, bicycle, car, van, truck, tricycle, awning-tricycle, bus, and motor.
2. **Normalization:** To calibrate the absolute pixel values, we adjusted the values with the normalized coordinates of the pixels (0-1), with respect to the image size.

4.2 Coordinate Conversion Logic (Code Snippet)

The following Python code was used to parse the raw annotations and convert them into the YOLO standard format.

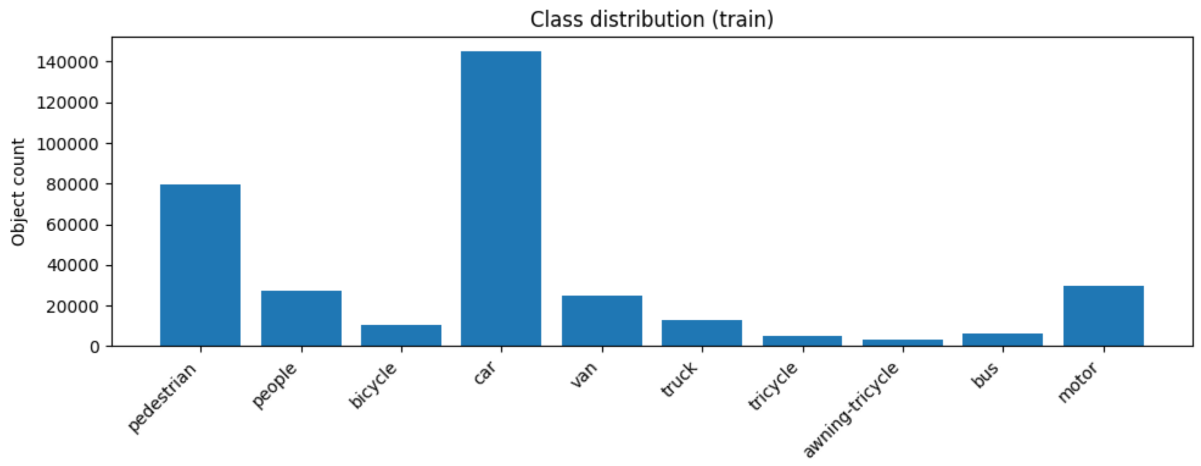
Python

```python
# Conversion Logic: VisDrone to YOLO Format
def convert_annotation(x, y, w, h, img_W, img_H):
  # Calculate normalized center coordinates
  xc = max(0, min(1, (x + w/2) / img_W))
  yc = max(0, min(1, (y + h/2) / img_H))

  # Calculate normalized width and height
  wn = max(0, min(1, w / img_W))
  hn = max(0, min(1, h / img_H))

  return xc, yc, wn, hn
```

This function is applied iteratively over the dataset to generate .txt label files

4.3 Data Distribution Analysis

Before training, we analyzed the class balance. The dataset is mostly trained to detect "Cars" and "Pedestrians," which is typical for urban environments.



5. Phase 4: Modeling (Techniques for Detection)

5.1 Technique Justification

For the detection and counting mechanism, we have used the **YOLOv8 (You Only Look Once)** architecture.

- **Real-time Processing:** YOLO is a single-stage detector, which is why it makes predictions of single-pass binding box and class probabilities. This renders it much quicker than two-stage detectors such as Faster R-CNN which can be used to process video in real-time.
- **Accuracy:** It is very useful in dealing with overlapping objects which is very vital in a crowded traffic scene.
- 5.2 Training Hyperparameters and Training Configuration.
- We optimized an existing yolov8m.pt (Medium) model. Accelerated convergence was done through transfer learning.
- **Epochs:** 20
- **Image Size:** 640x640
- **Optimizer:** SGD (Stochastic Gradient Descent)
- **Batch Size:** 16

5.3 Model Training Implementation (Code Snippet)

The following code shows how we started and trained the model on the existing dataset.

```
Python
from ultralytics import YOLO

# Load the pre-trained YOLOv8 Medium model
model = YOLO("yolov8m.pt")

# Train the model on the traffic dataset
model.train(

data="/content/drive/MyDrive/Apex_Traffic_Intelligence/models/configs/visdrone.yaml",
  epochs=20,
  imgsz=640,
  batch=16,
  project="/content/drive/MyDrive/Apex_Traffic_Intelligence/models/finetuned",
  name="apex_traffic_v1",
  save=True
)
```

5.4 Evaluation Results

After 20 epochs, the model achieved an **mAP@50 (Mean Average Precision) of 42.5%**.

- **Car Detection:** High precision (~73%), indicating highly reliable vehicle counting.
- **Truck/Bus Detection:** Moderate precision, acceptable for general traffic density estimation.

---

6. Phase 5: Deployment (Counting Techniques)

6.1 Inference and Counting Logic

Once the model detects objects, the system must "count" them to generate useful data. We utilize a frame-by-frame inference loop.

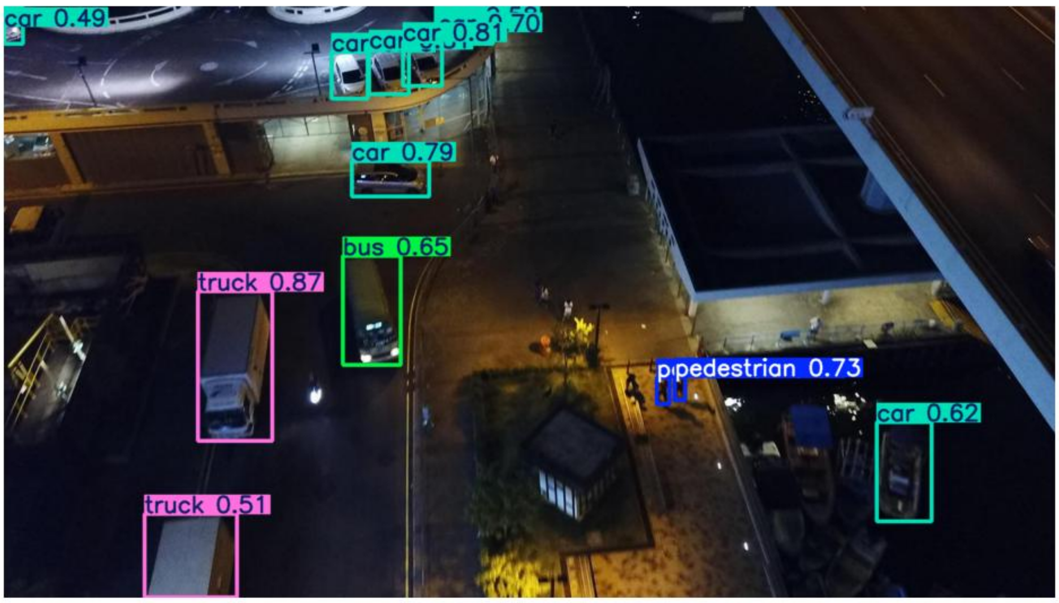
1. **Prediction:** The model makes predictions of bounding boxes of a frame.
2. **Filtering:** We use confidence threshold (e.g. 0.4) to remove weak prediction.
3. **Aggregation:** As we run over the identified boxes, we count with each type of vehicle.

6.2 Detection Implementation (Code Snippet)

The code below performs inference on a new image and counts the vehicle types present.

```python
from collections import Counter

# Run inference on a test image
results = model.predict(source_img_path, conf=0.4)
result = results[0]

# Initialize counter
class_counts = Counter()

# Iterate through detections to count vehicles
for box in result.boxes:
   cls_id = int(box.cls[0])
   class_name = result.names[cls_id]
   class_counts[class_name] += 1

print("Traffic Count:", class_counts)
```

Vehicle Detection Results



7. Phase 6: Reporting and Visualization

7.1 Automated Dashboarding

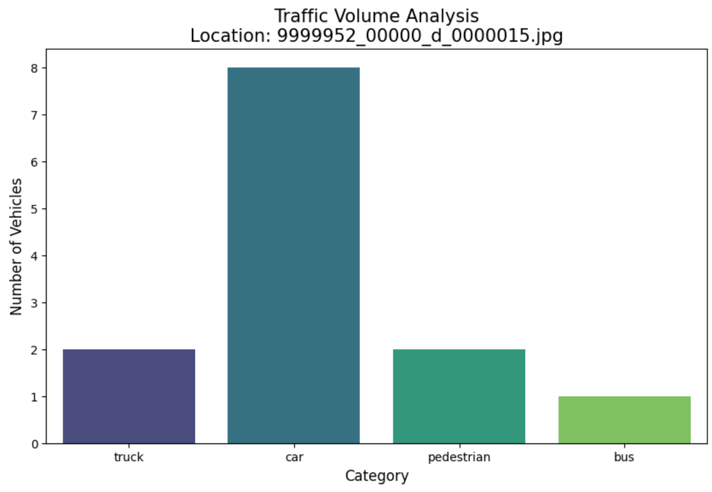
Information can only be of use when it is visualized to the decision makers. Apex Research will provide automated dashboards generated via Python.

- **Traffic Volume Snapshot:** A breakdown of vehicle types at a specific timestamp.
- **Congestion Alerts:** Triggered if the "Car" count exceeds a specific threshold.

7.2 Report Generation Implementation (Code Snippet)

We use matplotlib and seaborn to generate visual reports automatically after analysis.

```
Python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Prepare data for the report
df_report = pd.DataFrame(list(class_counts.items()), columns=['Vehicle Type', 'Count'])

# Generate Traffic Volume Bar Chart
plt.figure(figsize=(10, 6))
sns.barplot(data=df_report, x='Vehicle Type', y='Count', palette='viridis')
plt.title('Real-time Traffic Volume Analysis')
plt.xlabel('Category')
plt.ylabel('Number of Vehicles')

# Save report for stakeholders
plt.savefig("traffic_report.png")
```



8. Conclusion

This project demonstrates that the Data Science Lifecycle can be effectively applied to transportation. The Apex Research has created a prototype which can perform automated data collection of traffic by cleaning raw drone data, training a YOLOv8 model and applying counting logic. This system will enable the client to deal with a challenge of congestion through data-driven insight instead of manual one.

9. References

1. **Ultralytics YOLOv8 Documentation.** (2023). *Ultralytics YOLO Docs*. Retrieved from https://docs.ultralytics.com
2. **VisDrone Dataset.** (2019). *VisDrone2019: The Vision Meets Drone Object Detection in Images Challenge Results*. Retrieved from http://aiskyeye.com/
3. **Redmon, J., et al.** (2016). *You Only Look Once: Unified, Real-Time Object Detection*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
4. **OpenCV Library.** (2024). *OpenCV Documentation*. Retrieved from https://opencv.org/