Vehicle Movement Analysis and Insight Generation Project

Problem Statement

Title: "Vehicle Movement Analysis and Insight Generation in a College Campus using Edge AI"

Objective:

The primary objective of this project is to develop an Edge AI-based solution that can analyse vehicle movement in and out of a college campus using data from cameras capturing vehicle photos and license plates. The solution should provide insights on vehicle movement patterns, parking occupancy, and match vehicles to an approved vehicle database.

Problem Description:

Managing vehicle movement and parking in a college campus can be a challenging task. An intelligent system that can analyse vehicle movement, monitor parking occupancy, and match vehicles to an approved database can significantly improve campus security and management.

1. Dataset Description

   Code:

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="28Dts4nbe2jIxkAXVO4K")
project = rf.workspace("sanskar-2tjnr").project("intel-pzad1")
version = project.version(2)
dataset = version.download("yolov8")
```

- This is a custom dataset in which we have used 5 different videos.
- By capturing different frames and annotating the respective desired object like number-plates, date and time.
- We used Roboflow for the annotation part
- And exported the dataset in YOLOv8 model using the above code

2. Methodology

- This project focuses on developing an Edge AI solution for vehicle movement analysis and insight generation within a college campus. The system aims to analyse vehicle movement patterns, monitor parking occupancy, and match vehicles against an approved database using camera data. It outlines steps for data collection, pre-processing, analysis, and insight generation. The solution will be evaluated based on accuracy, real-time processing efficiency, and its ability to provide insights on vehicle movement, parking occupancy, and vehicle matching.
- Our task is to develop an Edge AI-based solution that can analyse vehicle movement using data from cameras capturing vehicle photos and license plates. The solution should be capable of processing image data in real-time and provide insights on:

    1. Vehicle Movement Patterns: Analyse the frequency and timing of vehicle movement in and out of the campus, identifying peak times and patterns.
    2. Parking Occupancy: Monitor the occupancy of parking lots in real-time, identifying which parking lots are frequently occupied and at what times
    3. Vehicle Matching: Match captured vehicle images and license plates to an approved vehicle database to identify unauthorized vehicles.

Tools and libraries used in the project:

1. Opencv-Python
2. Python
3. Pandas
4. Matplotlib
5. Seaborn
6. Roboflow
7. Yolov8
8. Ultralytics
9. Yolov8
10. Google Collab

Detailed Explanation of Main code:

1. **Libraries Used:**
    - OS: Provides a way to interact with the operating system (used for path manipulation in this case).
    - Pandas: A powerful library for data analysis and manipulation.
    - OpenCV: a computer vision library used for image and video processing.
    - Ultralytics: A library for loading and using YOLO (You Only Look Once) object detection models.
    - Easyocr: A library for optical character recognition (OCR), used to extract text from images.

2. **Load Model and OCR Reader:**
    - The script loads a YOLOv8 model from the specified path (model_path). This model is trained to detect objects (in this case, license plates) in images.
    - An EasyOCR reader is initialized to recognize text in English.
3. **Predict on Video:**
    - model. Predict(source=video path, conf=0.25, save=True) is used to run inference on the video specified by video path.
        - conf=0.25 sets a confidence threshold of 25%. Only detections with confidence scores above this threshold will be considered.

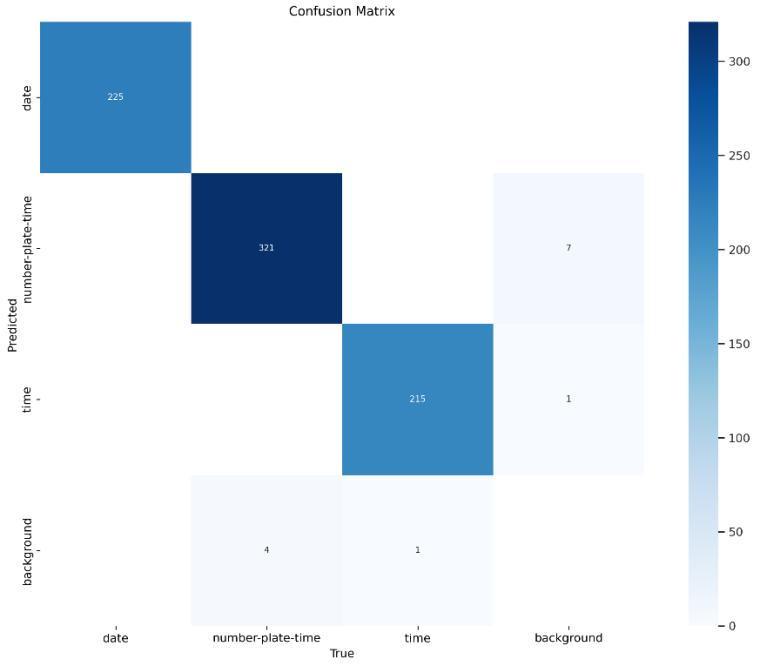
Initialize Frame Data:

- A dictionary frame data is created to store the extracted data for each frame. It is initialized with empty strings for the categories 'date', 'time', and 'number plate'.

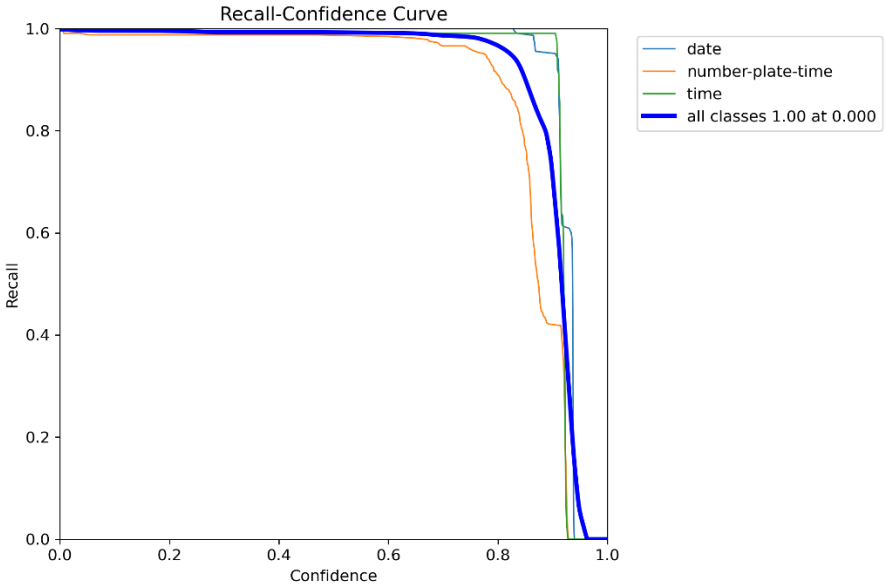
4. **Define classify text Function:**
    - This function takes the detected text as input and attempts to classify it into one of the categories:
        - If the text contains "date" or date-like delimiters ('/', '-'), it's classified as a 'date'.
        - If the text contains "time" or a colon (':'), it's classified as a 'time'.
        - Otherwise, it's classified as a 'numberplate'.
5. **Process the First Frame with Detections:**
    - The script iterates through the results of the YOLO predictions.
    - It extracts the original frame (result.orig_img) and iterates over the detected bounding boxes (result. Boxes).
    - For each box:
        - It extracts the bounding box coordinates (xmin, ymin, xmax, ymax) and the confidence score.
        - It crops the detected region from the original frame.
        - EasyOCR is used to perform text detection on the cropped image.
        - The detected text is joined into a single string (detected text).
        - The classify text function classifies the detected text into the appropriate category.
        - The classified text is stored in the frame_data dictionary.
    - The loop breaks as soon as at least one of the categories has been filled with a value.
6. **Save Results to Excel:**
    - The extracted data is converted into a Pandas DataFrame.
    - The DataFrame is saved to an Excel file (output path).

3. Results and Discussion

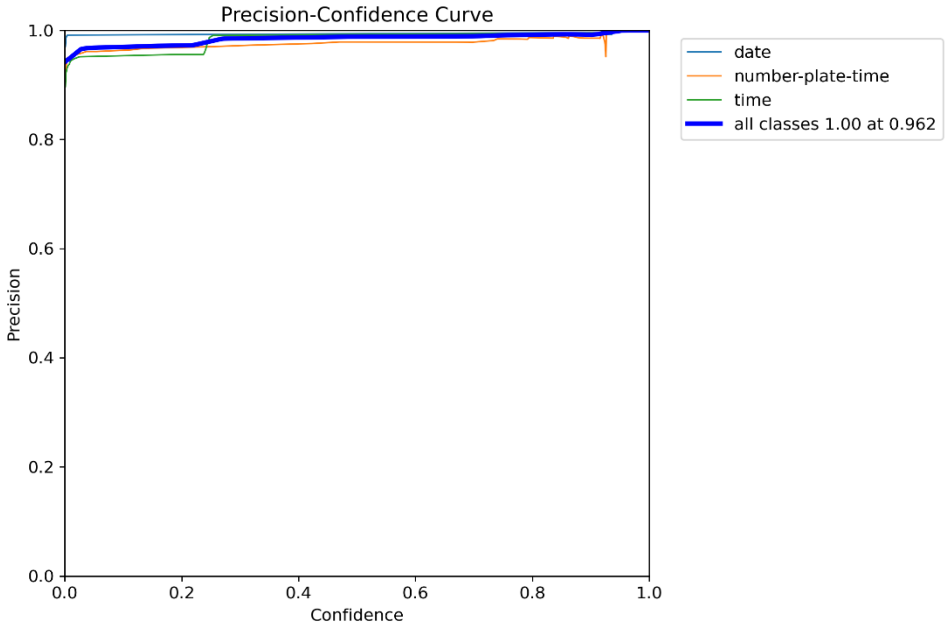
Confusion Matrix for the Yolo Model



R-Curve



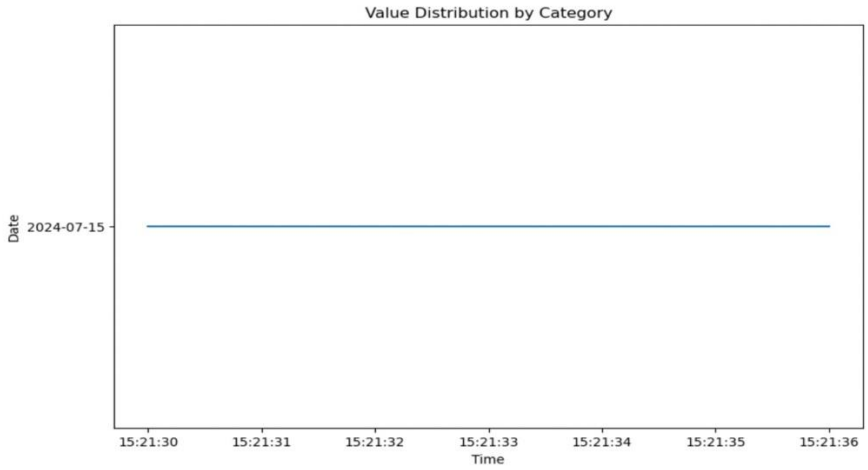
P-Curve



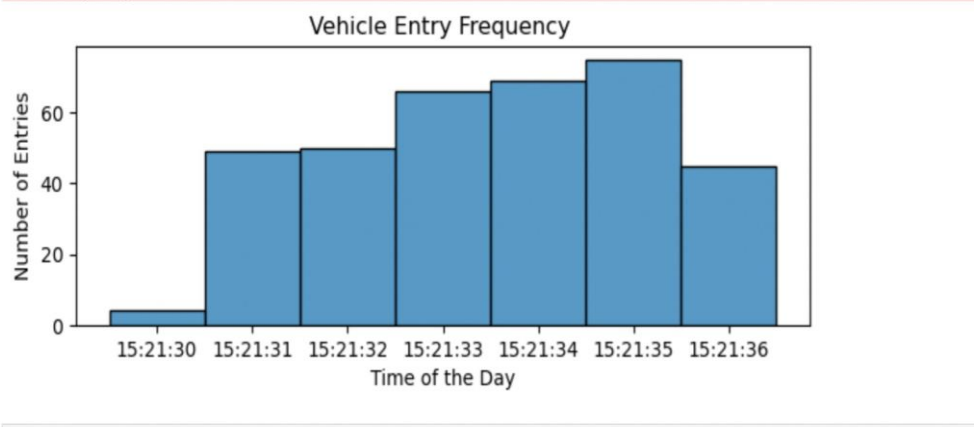
Model Detection Snap:

Results of the Data Analysis:

Line Plot



Histogram Plot



Output in Excel Sheet:

umber Pla	Date	Time
UP32HA17	2024-07-1	15:21:30
KA20U802	2024-07-1	15:21:30
17.06.03	2024-07-1	15:21:30
19/05/201	2024-07-1	15:21:30

Key Points and Potential Improvements:

- **Robustness:** Real-world implementations would need more robust handling of cases where text detection fails or the classification is ambiguous.
- **Efficiency:** You could potentially optimize the text detection and classification steps for faster processing.
- **Scalability:** Consider how to process large volumes of video data, perhaps by distributing the workload or using cloud resources for more computationally intensive tasks.
- **Flexibility:** Design the system to be easily adaptable to different camera positions, lighting conditions, and types of license plates.
- **Error Handling:** Implement proper error handling mechanisms to address potential exceptions during video processing, object detection, or text recognition.
- **Proper Dataset:** Due to insufficient data, we were unable to get the vehicles outing time.

4. Conclusion:

The application of computer vision and deep learning techniques in vehicle movement analysis has demonstrated significant potential across various fields. By harnessing these advanced technologies, we have been able to achieve precise and efficient detection, tracking, and analysis of vehicles in real-world scenarios.

5.Contribution Report:

Name of the team Members:

- Sanskar Srivastava (Team Leader)
- Aan Sikka

Individual Contribution:

- Sanskar Srivastava- Worked on the main flow of the code using OpenCV and OCR technique to extract the number plate characters. Annotated the images using Roboflow.
- Aan Sikka-Worked on the annotation of the images using Roboflow and trained the model using YOLOv8 and performing exploratory data analysis to generate insights on the output of the model.
- Both worked together in the ideation, research and overall formation of the project. Created the report and ppt also together.

Challenges Faced:

- As we both are from electrical engineering backgrounds, we learned all the technologies used along the way like OpenCV and YOLOv8 under the Intel's mentor guidance.

- Finding and creation of dataset was a major challenge as annotating so many images is extremely time consuming and exhausting.