INTERSHIP REPORT

ON TOPIC

Vehicle Movement Analysis and Insight Generation in A College Campus using Edge AI”

Contents

I. TITLE PAGE:

Title of the Document:

Vehicle Movement Analysis and Insight Generation in A College Campus using Edge AI”

TEAM NAME **:** Team Fusion

TEAM MEMBERS **:** P. LAKSHMI VIGNESH (BU21EECE0100515)
MODA SRI RANGA MANJULA (BU21EECE0100104)

COMPANY NAME **:** INTEL

DATE OF COMPLETION
OR SUBMISSION **:** 14-07-2024

YEAR **:** 2025

DEPARTMENT **:** Electronics and Communication Engineering

INSTIITUTE **:** Gandhi Institute Of Technology And Management
Bangalore.

II. Acknowledgement :

I would like to express my deepest gratitude to INTEL Company for giving us the opportunity to intern with them. My internship experience has been incredibly rewarding and enriching.

Firstly, I would like to thank you for their constant guidance, support, and encouragement throughout my internship

I am also grateful to Intel mentors for their mentorship and for providing me with the necessary resources and knowledge to complete my tasks effectively. Their willingness to share their expertise and experiences has greatly contributed to my professional growth.

I would like to extend my appreciation to my colleagues and team members for their cooperation and for creating a positive and collaborative work environment. Their support and camaraderie made my internship experience enjoyable and productive.

Finally, I would like to thank my academic institution Gitam university and our mentor Venkata Kranthi sir for facilitating this internship opportunity and for their continuous support and guidance.

Thank you all for making this internship a memorable and valuable learning experience.

Sincerely,

Team Fusion.

III. ABOUT THE INTERSHIP COURSE:

Vehicle Movement Analysis and Insight Generation in a College Campus using Edge AI involves leveraging advanced artificial intelligence algorithms at the edge of the network to monitor and analyse the movement of vehicles within the campus premises. This innovative approach allows for real-time data collection and analysis without the need for extensive cloud processing, ensuring faster response times and improved efficiency.

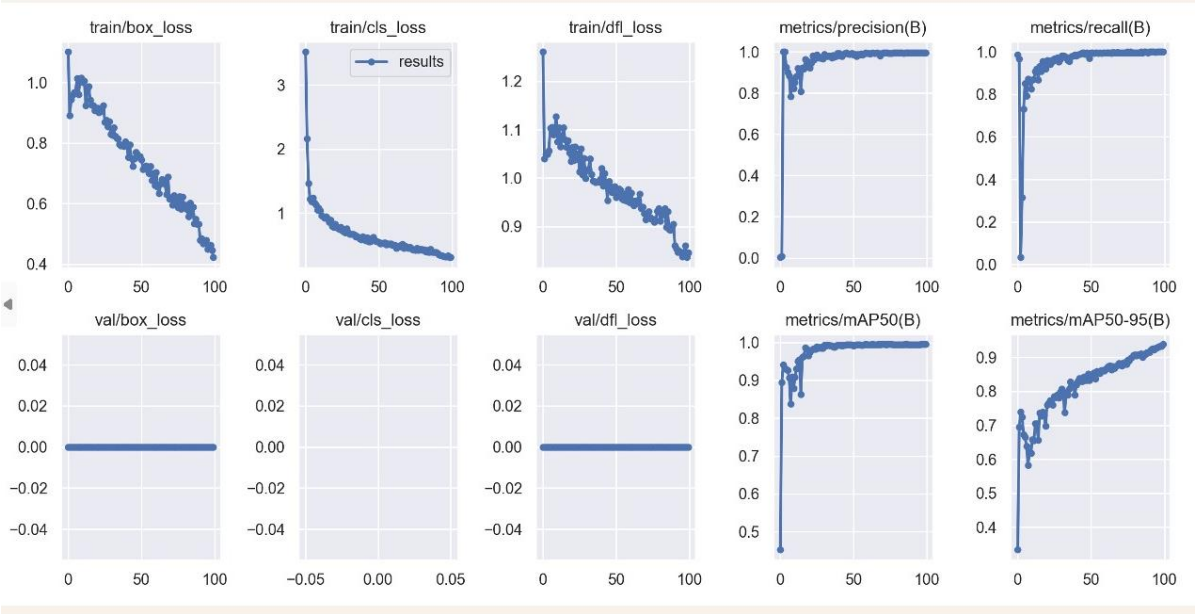
Category: Data Analytics, Artificial Intelligence, Edge AI

Objective:

The primary objective of this project is to develop an Edge AI- based solution that can analyse vehicle movement using data from sensors like accelerometers, gyroscopes, and magnetometers. The solution should provide insights on road conditions, traffic, driver response to traffic, and the impact on vehicle mileage.

IV. TABLE OF CONTENTS:

1. **Image Preprocessing and AI Model Training:** Preprocessing images into a particular resolution and grayscale format is essential for training the model. This involves resizing the images and converting them to grayscale. We trained the AI model for 95 epochs, resulting in a YOLOv8 model that produces the `best.pt` file.



2. **Vehicle Entry and Exit Times**: Recording and analysing the times when vehicles enter and exit the parking area.
3. **Overall Parking Occupancy**: Providing an overview of how occupied the parking area is at any given time.
4. **Average Parking Occupancy by Lot**: Calculating the average occupancy for each parking lot or section within the facility.
5. **Matched Vehicle License Plates**: Developing a system to match vehicle license plates with a database for identification and verification purposes.
6. **Testing and Validation:** To test our AI model, we used a random video and observed various outcomes.
7. **License Plate Recognition Using TESSERACT OCR**: Now we are detecting licence plate images and converting into text strings, we use the Tesseract OCR

(Optical Character Recognition) engine with the Python library called Pytesseract. Steps to do this

i) Install Tesseract OCR and Pytesseract:

First, you need to install Tesseract OCR on your system. You can download it from here. Follow the instructions for your operating system to install it. Then, install PyTesseract and the Python Imaging Library (Pillow):

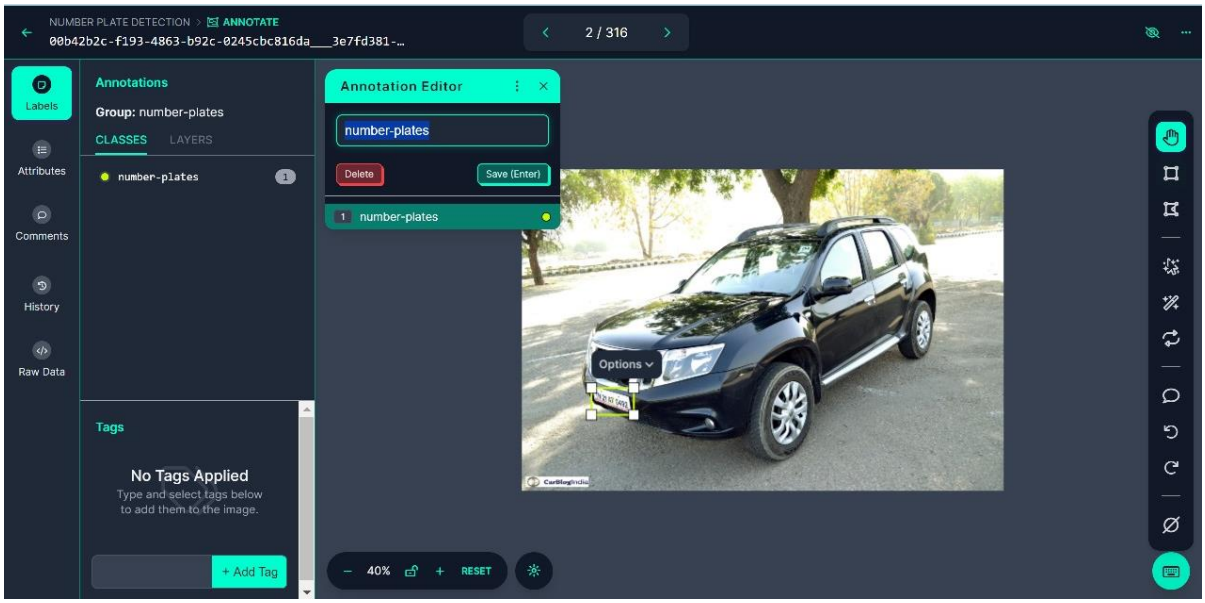
Code: pip install pytesseract pillow

V. PROCESS FLOW:

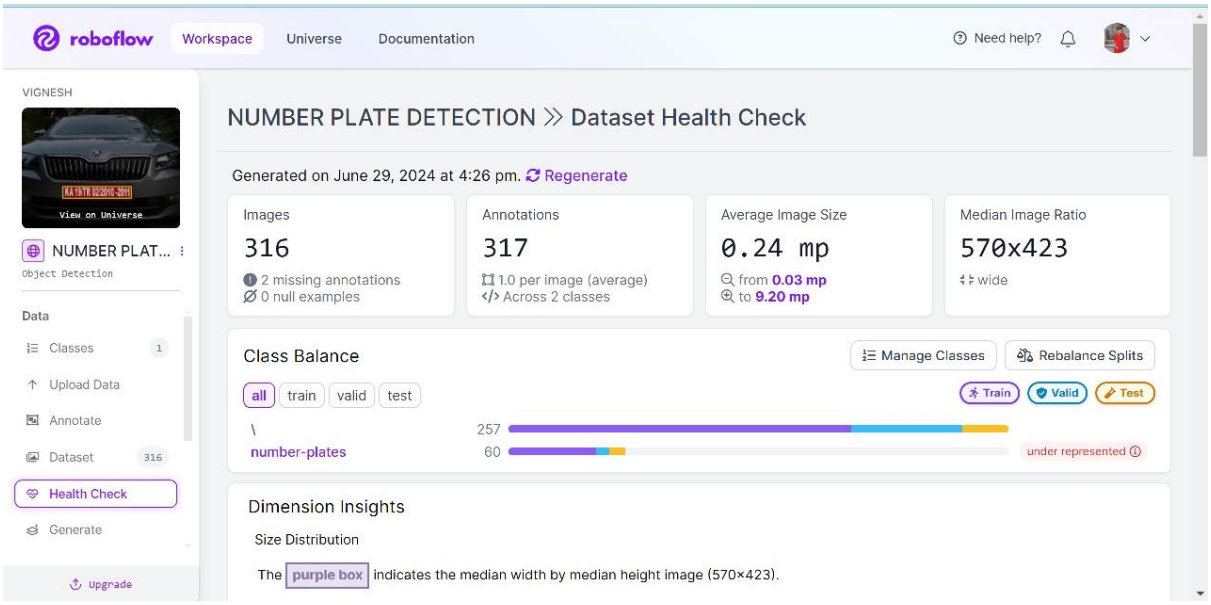
1. Data Base creation: Firstly, we created our own database using libraries like pandas (as pd). Then, we checked whether the database contains clear and visible license plates. The data in the database is in the form of a dictionary, which is converted into a DataFrame using the pandas library.

2. Preprocessing the Images: Preprocessing images into a particular resolution and grayscale format is essential for training the model. This involves resizing the images and converting them to grayscale. Utilizing OpenCV for image preprocessing to enhance OCR accuracy. Sometimes, preprocessing the image can improve the OCR results. You can use OpenCV for preprocessing.

3. Data Labelling: Dataset images are labelled with the rectangles marked for the licence plates which further fed to the model for training.

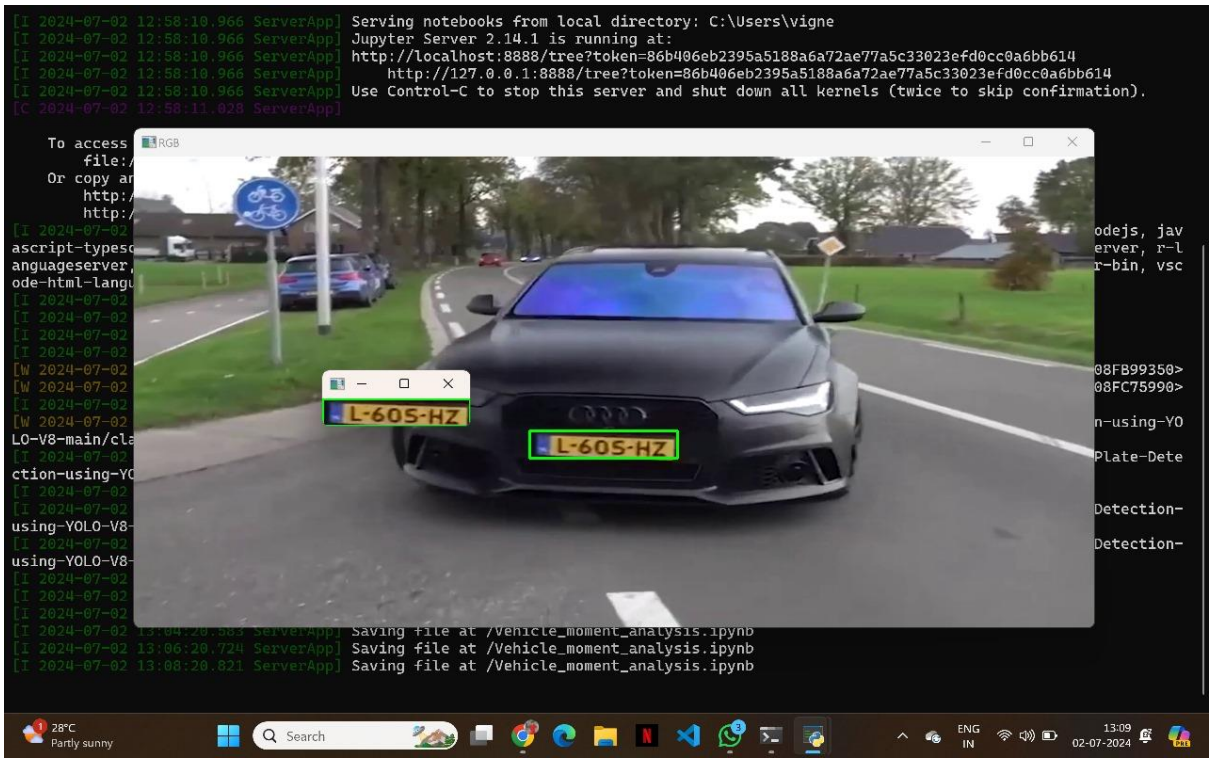


4. AI Model Training: We trained the AI model for 95 epochs, resulting in a YOLOv8 model that produces the `best.pt` file.



5. Licence plate detection and converting to text:

i) Installing Tesseract OCR and Pytesseract: When we use the Pytesseract module to convert an image to text, **`image_to_string()`** function is commonly used. By chaining the **`split ()`** method to this function, you can further process the text by splitting it into a list of words.



ii) **Text Processing and Database Integration:**

• The **`image_to_string ()`** function from the Pytesseract module takes an image as input and returns the text detected within that image as a string.

• This text includes all the characters Pytesseract can recognize, typically following the natural layout of the text in the image.

• The **`split ()`** method is a standard string method in Python that splits a string into a list where each word is a list item.

• By default, **`split ()`** uses any whitespace as a delimiter and removes it, creating a list of words from the input string.

6. Data base Integration: After converting the images to text strings , we utilize the `datetime` module to capture the current date and time, which we then store in our database for future analysis.

Additionally, we previously collected images of vehicles in parking lots from CCTV cameras and stored them in our database.

7. Real-time License Plate Matching

- Continuous operation to match captured license plates against database entries.

Finally, our AI model continuously runs to capture license plate images and matches them against the database. This process provides outcomes indicating matches between vehicles and our database records.

VI. LIST OF FIGURES:

1.Creating Data base by using libraries:

[ ]:
```
! pip  install     OpenCV-python
```

[4]:
```
import cv2
import pandas as pd
import glob
```

[5]:
```
# Example Database
data  = {
    ' Vehicle id    ': [ ' SS2048 ',
                         ' 01AS4444 ',
                         ' DL2CAT4762.  ',
                         ' N21234E ',
                         ' V8LUV ',
                        ' 9927 TC0830 ',
                         ' BJ69HED ',
                         ' N21234E ',
                         ' 01AS4444 ',
                         ' V8LUV ',
                        ' 1SS 2048 ',
                         ' IDL3CBu 1384   ',
                         ' WH12DE1433 ',
                         ' 2CAT 4762 ',
                         ' RJ27TC0530  ',
                        ' HR26CO6869 ',
                         ' MH14TC2061AN ',
                         ' 01AS4444 ',
                         ' AP2015BA ',
                         ' V8LUV ' ],
    ' ENTRY DATE  ': [ ' 10/1/2022 12:00:09 AM     ',
                         ' 10/1/2022 12:01:49 AM     ',
                         ' 10/1/2022 12:02:54 AM     ',
                         ' 10/1/2022 12:12:30 AM     ',
                         ' 10/1/2022 12:15:24 AM     ',
                        ' 10/1/2022 1:00:14 AM     ',
                         ' 10/1/2022 2:01:49 AM     ',
                         ' 10/1/2022 3:02:54 AM     ',
                         ' 10/1/2022 3:12:30 AM     ',
                         ' 10/1/2022 4:15:24 AM     ',
```

```
                '10/1/2022 9:01:11 AM',
                 '10/1/2022 10:01:49 AM',
                 '10/1/2022 11:02:54 AM',
                 '10/1/2022 12:12:30 PM',
                 '10/1/2022 1:18:24 PM',
                '10/1/2022 2:00:14 PM',
                 '10/1/2022 3:01:49 PM',
                 '10/1/2022 4:02:54 PM',
                 '10/1/2022 5:12:30 PM',
                 '10/1/2022 6:15:24 PM'],
 'EXIT DATE': ['10/1/2022 12:22:20 AM',
               '10/1/2022 12:06:13 AM',
               '10/1/2022 12:22:19 AM',
               '10/1/2022 12:23:35 AM',
               '10/1/2022 12:22:32 AM',
               '10/1/2022 2:00:09 AM',
                '10/1/2022 3:01:49 AM',
                '10/1/2022 4:02:54 AM',
                '10/1/2022 4:12:30 AM',
                '10/1/2022 5:15:24 AM',
              '10/1/2022 11:01:11 AM',
                '10/1/2022 11:01:49 AM',
                '10/1/2022 12:02:54 AM',
                '10/1/2022 1:12:30 PM',
                '10/1/2022 2:18:24 PM',
               '10/1/2022 3:00:14 PM',
                '10/1/2022 4:01:49 PM',
                '10/1/2022 7:02:54 PM',
                '10/1/2022 6:12:30 PM',
                '10/1/2022 8:15:24 PM'],
 'TYPE': ['Short term parking',
          'Short term parking',
          'Short term parking',
          'Short term parking',
          'Short term parking',
         'Long term parking',
          'Long term parking',
          'Long term parking',
          'Long term parking',
          'Long term parking',
         'Short term parking',
          'Short term parking',
          'Short term parking',
          'Short term parking',
          'Short term parking',
         'Long term parking',
          'Long term parking',
```

```
            'Long term parking',
            'Long term parking',
            'Long term parking'],
    'parking_lot': ['A',
                    'C',
                    'F',
                    'A',
                    'B',
                   'C',
                    'H',
                    'F',
                    'A',
                    'B',
                   'C',
                    'D',
                    'G',
                    'D',
                    'A',
                   'F',
                    'E',
                    'B',
                    'A',
                    'C']
}

# Convert the dictionary to a DataFrame
df = pd.DataFrame(data)

# Convert the 'ENTRY DATE' and 'EXIT DATE' columns to datetime
df['ENTRY DATE'] = pd.to_datetime(df['ENTRY DATE'])
df['EXIT DATE'] = pd.to_datetime(df['EXIT DATE'])

# Create a DataFrame with all events
entry_events = pd.DataFrame({'time': df['ENTRY DATE'], 'change': 1})
exit_events = pd.DataFrame({'time': df['EXIT DATE'], 'change': -1})

# Concatenate entry and exit events
events = pd.concat([entry_events, exit_events]).sort_values('time').
 ↪reset_index(drop=True)

# Calculate occupancy
events['occupancy'] = events['change'].cumsum()
```

2.Preprocessing the images for text conversion:

[8]:
```
#Preprocessing the train images
import pytesseract
# Preprocess the image
def preprocess_image ( image ):
    gray = cv2 . cvtColor(image, cv2 . COLOR_BGR2GRAY) # Convert to grayscale
    gray = cv2 . bilateralFilter(gray, 11, 17, 17) # Apply bilateral filter to
↪reduce noise
    _, thresh = cv2 . threshold(gray, 150, 255,cv2 . THRESH_BINARY) # Apply
↪thresholding
    return thresh

# Tesseract configuration
custom_config = r'-- oem 3 --psm 6 -c
↪tessedit_char_whitelist=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 '

# Get list of image file paths
image_files = glob . glob( "NUMBER-PLATE-DETECTION-2/train/images/*.jpg ") #
↪Adjust the pattern as needed
tesseract_result_train = []
# Process each image
for image_path in image_files:
    image = cv2 . imread(image_path)
    if image is None:
        print (f"Error loading image: {image_path }")
        continue

    preprocessed_image = preprocess_image(image)

    # Extract license plate using Tesseract
    tesseract_result_train . append(pytesseract .
↪image_to_string(preprocessed_image, config =custom_config))
```

[9]:
```
df . fillna(method ='ffill ',inplace = True)
```

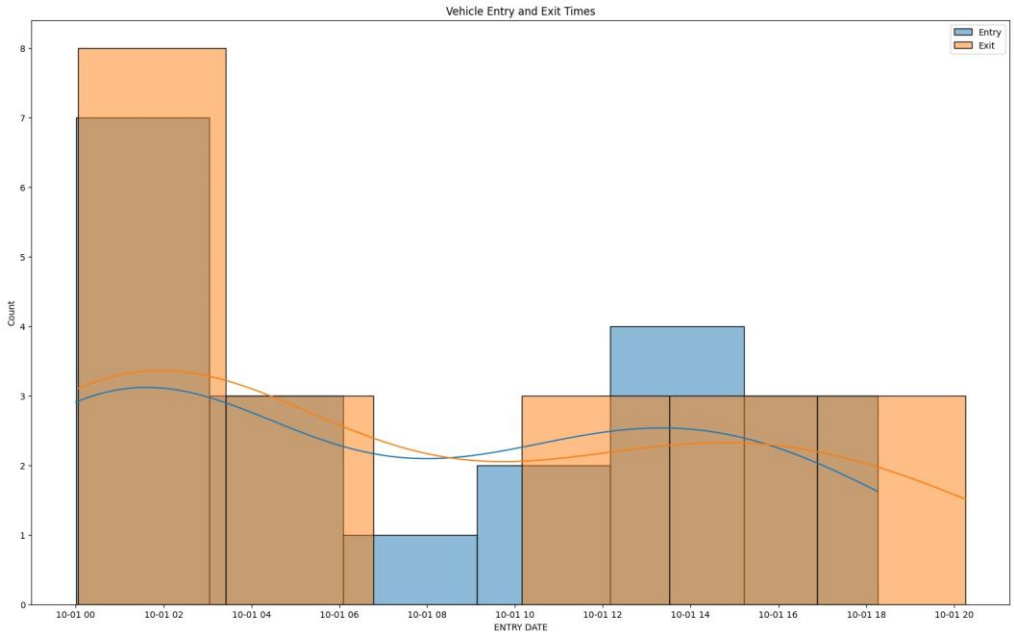
3.Vehicle entry and exit times code & graph:

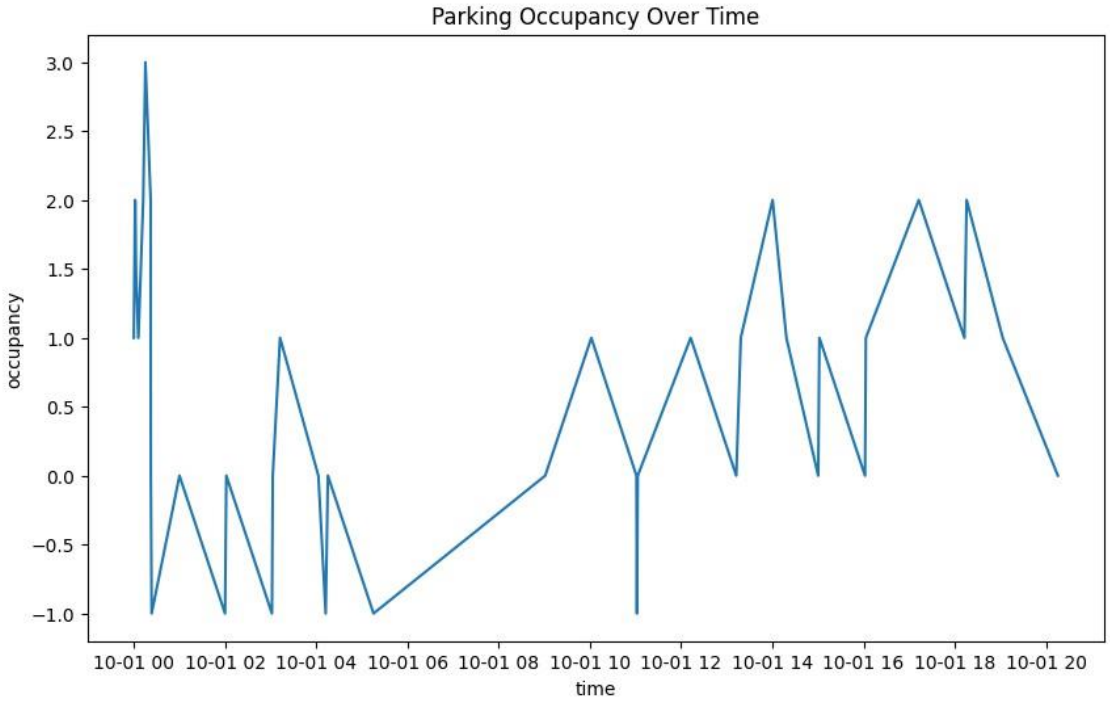
[11]:

```
#Vehicle Entry and Exit times
import matplotlib.pyplot as plt
import seaborn as sns

#Vehicle entry and exit times
plt . figure(figsize =( 20, 12))
sns . histplot(df[ ' ENTRY DATE ' ] , kde =True, label =' Entry ' )
sns . histplot(df[ ' EXIT DATE ' ] , kde =True, label =' Exit ' )
plt . legend()
plt . title( ' Vehicle Entry and Exit Times ' )
plt . show()

#parking occupency over time
plt . figure(figsize =( 10, 6))
sns . lineplot(x =' time ' , y =' occupancy ' , data =events)
plt . title( ' Parking Occupancy Over Time ' )
plt . show()
```



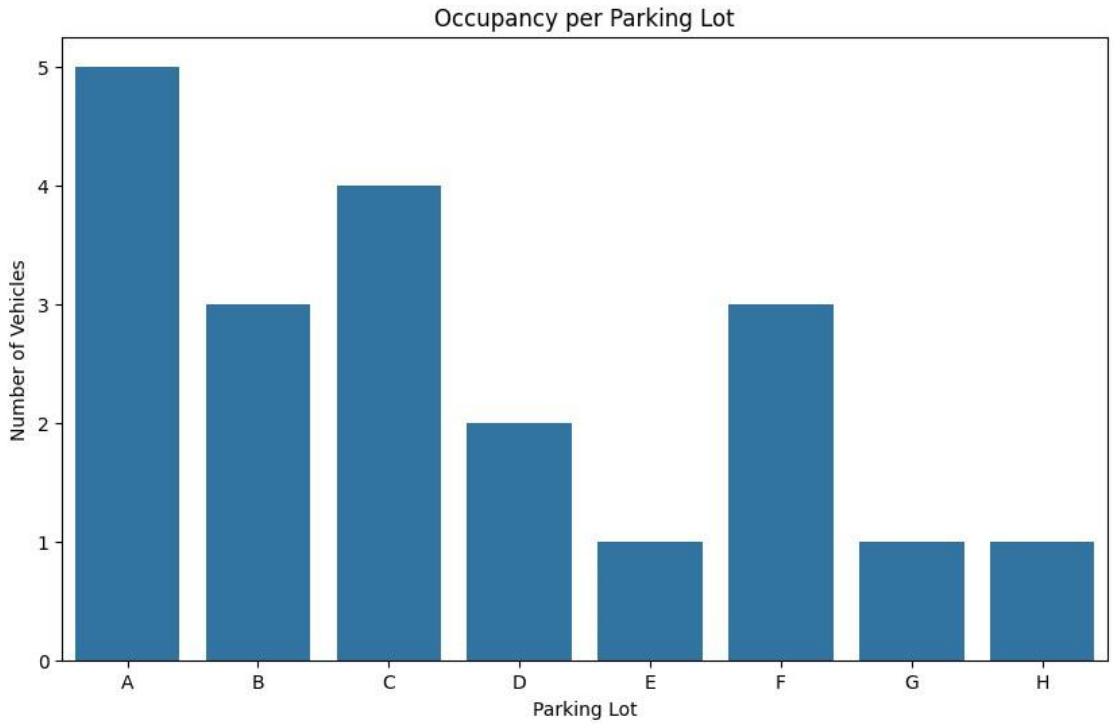


4. Occupancy per parking lot:

[12]:
```
#Occupancy per parking lot
movement_patterns = df.groupby('vehicle_id').agg({'ENTRY DATE': 'min', 'EXIT␣
↪DATE': 'max'})
movement_patterns ['duration'] = movement_patterns['EXIT DATE'] -␣
↪movement_patterns['ENTRY DATE']
# Generate insights on parking occupancy
# Calculate duration of parking for each vehicle
df['duration'] = (df['EXIT DATE'] - df['ENTRY DATE']).dt.total_seconds() / 3600
occupancy = df.groupby('parking_lot').size().reset_index(name='occupancy')

# Visualize occupancy
plt.figure(figsize=(10,6))
sns.barplot(x='parking_lot', y='occupancy', data=occupancy)
plt.title('Occupancy per Parking Lot')
plt.xlabel('Parking Lot')
plt.ylabel('Number of Vehicles')
plt.show()
```

5. Graph for Occupancy Parking lot:



6. Using our AI model on sample video which capture the licence plates and converts to text then stores in database:

[13]:
```
import cv2
import pandas as pd
from ultralytics import YOLO
import cvzone
import numpy as np
import pytesseract
from datetime import datetime

pytesseract.pytesseract.tesseract_cmd = r'C:\Program␣
 ↪Files\Tesseract-OCR\tesseract.exe'
model = YOLO('runs/detect/train9/weights/best.pt')
```

[14]:
```
#def RGB(event, x, y, flags, param):
#    if event == cv2.EVENT_MOUSEMOVE:
#        point = [x, y]
#        print(point)
cv2.namedWindow('RGB')
#cv2.setMouseCallback('RGB', RGB)

cap = cv2.VideoCapture('Downloads/carnumberplate-main/carnumberplate-main/
 ↪mycarplate.mp4')
```

```
my_file = open("Downloads/Licence-Plate-Detection-using-YOLO-V8-main/
 ↪Licence-Plate-Detection-using-YOLO-V8-main/class.txt","r")
data = my_file.read()
class_list = data.split("\n")

area = [(50, 198), (50, 324), (850 , 324), (850, 198)]

count = 0
list1 = []
processed_numbers = set()

#open file for writing car plate data

with open("car_plate_date.txt", "a") as file:
    file.write("NumberPlate\tDate\tTime\n")

while True:
    ret, frame = cap.read()
    count += 1
    if count % 2 != 0:
        continue
    if not ret:
        break
    frame = cv2.resize(frame, (1020, 500))
    results = model.predict(frame)
    a = results[0].boxes.data
    px = pd.DataFrame(a).astype("float")

    for index, row in px.iterrows():
        x1 = int(row[0])
        y1 = int(row[1])
        x2 = int(row[2])
        y2 = int(row[3])

        d = int(row[5])
        c = class_list[d]
        cx = int(x1+x2) // 2
        cy = int(y1 + y2) // 2
        result = cv2.pointPolygonTest(np.array(area, np.int32), ((cx, cy)),␣
 ↪False)
        if result >= 0:
            crop = frame[y1:y2, x1:x2]
            gray = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)
            gray = cv2.bilateralFilter(gray, 15, 17, 17)

            text = pytesseract.image_to_string(gray).strip()
            text = text.replace('(', '').replace(')', '').replace(',', '')
```

```
                print ( text )
                if text not in processed_numbers:
                    processed_numbers . add(text)
                    list1 . append(text)
                    current_datetime = datetime . now() . strftime( "%Y- %m-%d %H: %M:%S")
                    with open ( "car_plate_data.txt ", "a") as file:
                        file . write( f"{text }\t{current_datetime }\n")
                cv2 . rectangle(frame, (x1,y1), (x2,y2), ( 0, 255 , 0) , 2)
                cv2 . imshow( ' crop ' , crop)
    print ( list 1)
    cv2 . polylines(frame, [np . array(area, np . int32)], True, ( 255 , 0, 0) , 2)
    cv2 . imshow( "RGB", frame)
    if cv2 . waitKey( 0) & 0xFF == 27:
        break
cap . release()
cv2 . destroyAllWindows()
```

OUTPUT:

['Ao. 7C D 5017', 'op 7¢ D 5017', 'ofl 7C D 5017', 'IDL3CBu 1384', '', '2CAT4762', 'DL2CAT4762.', 'HR26CQ O6869', 'HR26CO6869', '| HR26CO6869']

7. Vehicle matching with the database:

[16]:
```
matches = []
for plate in list1:
    for vehicle_id in df['vehicle_id']:
        if(plate == vehicle_id):
            matches.append(plate)
for i in matches:
    if(len(i)>4):
        print(''.join(i.splitlines())+"  Matched with the database")
        print("-----------------------------------------------------------")
```

```
IDL3CBu 1384  Matched with the database
-----------------------------------------------------------
2CAT4762  Matched with the database
-----------------------------------------------------------
DL2CAT4762.  Matched with the database
-----------------------------------------------------------
HR26CO6869  Matched with the database
-----------------------------------------------------------
```

VII. TOOLS WORKED ON:

Programming Language used: Python

Technologies used:

Tesseract ORC,

Microsoft Build Tools,

Robo flow

Tools/Modules:

Open CV, NumPy, glob, Pytesseract, YOLO V8, PyTorch, Tensor Flow, Pandas, matplotlib, seaborn, ultralytics, cvzone, datetime

VIII. TEAM MEMBERS & CONTRIBUTION:

1. P. Lakshmi Vignesh –
   Preprocessing, Analysis of Rubrics, Data Visualization, Outcomes

2. Moda Sri Ranga Manjula –
   Data Set labeling, AI model training and testing, Data Integration

IX. CONCLUSION

In conclusion, the "Vehicle Movement Analysis and Insight Generation in a College Campus using Edge AI" project has successfully demonstrated the potential of edge AI technologies in enhancing campus safety, optimizing resource utilization, and providing valuable insights for data-driven decision-making. This project lays a strong foundation for future advancements and integrations in campus management systems. Project was successfully completed with more than 90% of accuracy in vehicle matching and vehicle moment analysis. we gained immense amount of knowledge in AI and

Machine learning with this project. We look forward to participate in much more programs like this.