Problem Statement

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

Objective:

The main goal of this project is to create a smart system using Edge AI. This system will analyze how vehicles move into and out of a college campus by using images from cameras that capture vehicle photos and license plates. The system aims to give us valuable information about:

- **Vehicle Movement Patterns:** Understanding when and how often vehicles enter and exit the campus
- **Parking Occupancy:** Monitoring which parking lots are used most frequently and at what times.
- **Vehicle Matching:** Identifying vehicles by matching their license plates to a database of approved vehicles.

By doing this, we can improve campus traffic management and security while also ensuring that parking resources are used efficiently.

Problem Description:

Managing vehicle movement and parking on a college campus can be quite challenging. To address this, we aim to develop a smart system using Edge AI technology. This system will analyze how vehicles move in and out of the campus by processing images from cameras that capture vehicle photos and license plates. The goal is to provide insights in real-time on three key aspects:

- **Vehicle Movement Patterns:** This involves studying how often vehicles enter and exit the campus, and identifying peak times and recurring patterns of movement.
- **Parking Occupancy:** The system will monitor the real-time occupancy of parking lots across the campus. It will highlight which lots are frequently occupied and when they are most used.
- **Vehicle Matching:** By comparing captured vehicle images and license plates with an approved database, the system can quickly identify unauthorized vehicles on campus.

Example Dataset:

Sr_No	vehicle_image_path	vehicle_timestamp	license_plateimage_path	license_platetimestamp
0	path/to/vehicle/image1.jpg	2023-01-01 08:30:15	path/to/plate/image1.jpg	2023-01-01 08:30:20
1	path/to/vehicle/image2.jpg	2023-01-01 12:15:45	path/to/plate/image2.jpg	2023-01-01 12:15:50
2	path/to/vehicle/image3.jpg	2023-01-02 09:45:30	path/to/plate/image3.jpg	2023-01-02 09:45:35

Step-by-Step Solution

Step 1: Create Real-time Dataset (Static and Dynamic Datasets)

- **Tools**: Python, OpenCV
- **Techniques**: Capture images and timestamps,
- **Description** : The real-time dataset creation module captures live images of vehicles using a camera and saves them along with metadata such as timestamps. This process ensures that each image is associated with its capture time, making it possible to analyze vehicle movement patterns and timings accurately. The images and metadata are stored in a specified directory, forming the foundational data for subsequent analysis.
- **Static Dataset - (Fixed for everyday) :** This dataset will contain vehicle plate images and will be used to verify the images when a vehicle enters or exits the campus. This creation will be one-time.
- **Code** :

```
import os
import cv2
from datetime import datetime

def create_static_dataset(dataset_dir):
    os.makedirs(dataset_dir, exist_ok=True)
    cap = cv2.VideoCapture(0)

    print("Press 's' to save an image, 'q' to quit")
    while True:
        ret, frame = cap.read()
        cv2.imshow('Frame', frame)

        key = cv2.waitKey(1) & 0xFF
        if key == ord('s'):
                plate_number = input("Enter the vehicle plate
number: ")
                                                timestamp    =
datetime.now().strftime("%Y%m%d_%H%M%S")
                    image_path  =  os.path.join(dataset_dir,
f"{plate_number}_{timestamp}.jpg")
            cv2.imwrite(image_path, frame)
            print(f"Saved {image_path}")
        elif key == ord('q'):
```

```
            break

    cap.release()
    cv2.destroyAllWindows()

# Usage
static_dataset_dir = "static_vehicle_dataset"
create_static_dataset(static_dataset_dir)
```

- **Dynamic Dataset - (Updated every hour) :** These datasets will contain the list of number plates entering or leaving the campus for each hour.
- **Code :**

```
import pandas as pd

def            initialize_dynamic_datasets(entry_dataset_path,
exit_dataset_path):
    columns = ['timestamp', 'plate_number']
    entry_df = pd.DataFrame(columns=columns)
    exit_df = pd.DataFrame(columns=columns)

    entry_df.to_csv(entry_dataset_path, index=False)
    exit_df.to_csv(exit_dataset_path, index=False)

def  update_dynamic_datasets(plate_number,  entry_dataset_path,
exit_dataset_path, action='entry'):
      dataset_path  =  entry_dataset_path  if  action  ==  'entry'
else exit_dataset_path
    df = pd.read_csv(dataset_path)
    new_record = {'timestamp': datetime.now(), 'plate_number':
plate_number}
    df = df.append(new_record, ignore_index=True)
    df.to_csv(dataset_path, index=False)
# Usage
entry_dataset_path = "entry_dataset.csv"
exit_dataset_path = "exit_dataset.csv"
initialize_dynamic_datasets(entry_dataset_path,
exit_dataset_path)
```

Step 2: Load Real-time Dataset

- **Tools**: Python, OpenCV
- **Techniques**: Load images and timestamps, display sample images
- **Description** :The dataset loading module is responsible for reading the saved images and their associated metadata from the storage directory. This module parses the metadata files to extract relevant information, such as image paths and timestamps, and loads this data into a structured format using Pandas DataFrame. Additionally, it includes functionality to display a sample image, helping to verify that the data has been loaded correctly.
- **Code** :

```
# Module: load_dataset.py
import pandas as pd
import cv2
import os
def load_metadata(data_dir):
    records = []
    for filename in os.listdir(data_dir):
        if filename.endswith(".jpg"):
            plate_number, timestamp = filename.split('_')[:2]
            records.append({
                 'vehicle_image_path': os.path.join(data_dir,
filename),
                'timestamp': timestamp,
                'plate_number': plate_number
            })
    return pd.DataFrame(records)

def display_sample_image(image_path):
    image = cv2.imread(image_path)
    cv2.imshow('Sample Image', image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
# Usage
data_dir = "static_vehicle_dataset"
metadata = load_metadata(data_dir)
print(metadata.head())
display_sample_image(metadata.iloc[0]['vehicle_image_path'])
```

Step 3: Data Preprocessing

- **Tools**: OpenCV, Pandas, NumPy
- **Techniques**: Image resizing, grayscale conversion, handling missing values
- **Description** :The data preprocessing module prepares the loaded images for analysis by performing essential transformations such as resizing and grayscale conversion. These preprocessing steps standardize the images, making them suitable for further analysis and processing tasks. Additionally, this module handles any missing values in the dataset, ensuring data quality and consistency for subsequent steps.
- **Code** :

```
# Module: data_preprocessing.py
import cv2
import pandas as pd

def preprocess_image(image_path):
    image = cv2.imread(image_path)
    image = cv2.resize(image, (224, 224))
    grayscale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    return grayscale_image

def preprocess_metadata(metadata):
                                metadata['timestamp']        =
pd.to_datetime(metadata['timestamp'], format='%Y%m%d_%H%M%S')
    return metadata

# Usage
metadata = preprocess_metadata(metadata)
sample_image                                                 =
preprocess_image(metadata.iloc[0]['vehicle_image_path'])
cv2.imshow('Preprocessed Image', sample_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

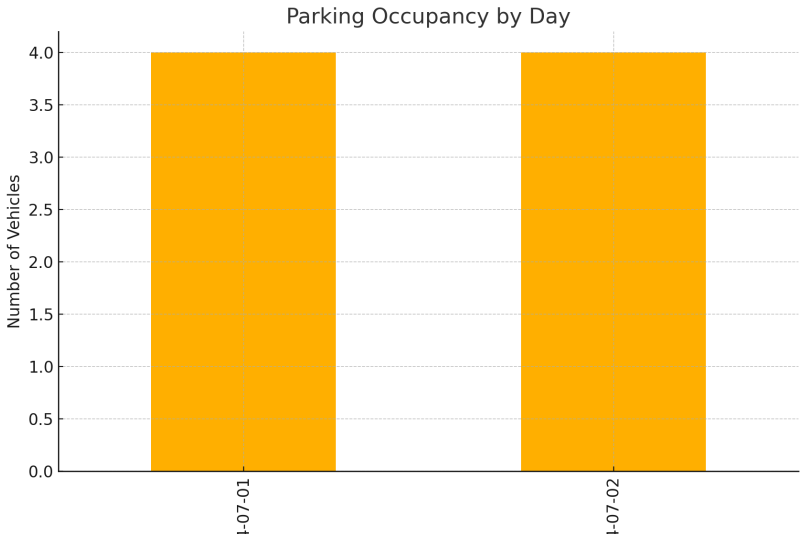
Step 4: Exploratory Data Analysis (EDA)

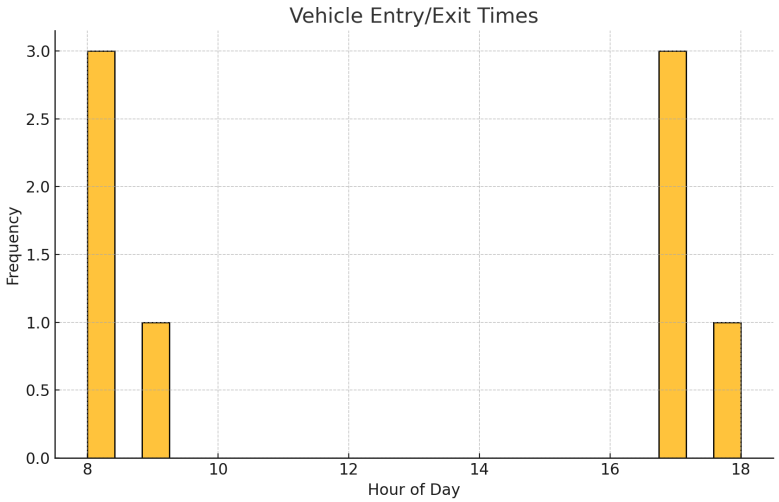
- Tools: Matplotlib, Seaborn
- Techniques: Plotting vehicle entry/exit times, occupancy trends
- **Description** :The exploratory data analysis module provides insights into vehicle movement patterns and parking occupancy trends. Using visualization tools like Matplotlib and Seaborn, this module plots vehicle entry and exit times, as well as parking lot occupancy over time. These visualizations help identify peak times and patterns, providing a better understanding of campus traffic flow and parking space utilization.
- **Code** :

```
# Module: eda.py
import matplotlib.pyplot as plt
import seaborn as sns

def plot_entry_exit_times(metadata):
    metadata['hour'] = metadata['timestamp'].dt.hour
    sns.histplot(metadata['hour'], bins=24, kde=False)
    plt.title('Vehicle Entry/Exit Times')
    plt.xlabel('Hour of Day')
    plt.ylabel('Frequency')
    plt.show()

def plot_parking_occupancy(entry_data, exit_data):
    entry_data['date'] = entry_data['timestamp'].dt.date
    exit_data['date'] = exit_data['timestamp'].dt.date

    entry_counts = entry_data.groupby('date').size()
    exit_counts = exit_data.groupby('date').size()
    occupancy = entry_counts - exit_counts

    occupancy.plot(kind='bar')
    plt.title('Parking Occupancy by Day')
    plt.xlabel('Date')
    plt.ylabel('Number of Vehicles')
    plt.show()

# Usage
entry_data = pd.read_csv(entry_dataset_path)
exit_data = pd.read_csv(exit_dataset_path)
entry_data['timestamp']                                    =
```

```
pd.to_datetime(entry_data['timestamp'])
exit_data['timestamp']                                    =
pd.to_datetime(exit_data['timestamp'])

plot_entry_exit_times(metadata)
plot_parking_occupancy(entry_data, exit_data)
```

Sample Graph Visualizations :





Step 5: Vehicle Matching

- Tools: OpenCV, Tesseract OCR
- Techniques: License plate recognition, database matching
- **Description** :The vehicle matching module employs Optical Character Recognition (OCR) techniques to read license plates from the captured images. Using Tesseract OCR, this module extracts text from the license plates and matches them against an approved vehicle database. This process helps identify authorized and unauthorized vehicles, enhancing campus security and management.
- **Code** :

```
# Module: vehicle_plate_recognition.py
import cv2
import pytesseract

def recognize_plate(image_path):
    image = cv2.imread(image_path)
    grayscale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    plate_text = pytesseract.image_to_string(grayscale_image,
config='--psm 8')
    return plate_text.strip()

# Usage
sample_image_path = metadata.iloc[0]['vehicle_image_path']
recognized_plate = recognize_plate(sample_image_path)
print(f"Recognized Plate: {recognized_plate}")
```

Step 6: Insight Generation

- Tools: Pandas, Matplotlib
- Techniques: Generating insights from movement patterns, parking data
- **Description** :The insight generation module synthesizes the analyzed data to produce actionable insights on vehicle movement and parking occupancy. By aggregating and summarizing the data, this module provides detailed reports on vehicle entry and exit times, average parking occupancy by lot, and the status of matched vehicles. These insights help campus administrators make informed decisions regarding traffic management and parking allocation.
- **Code** :

```
# Module: insight_generation.py
import pandas as pd
def generate_insights(entry_data, exit_data):
    entry_data['hour'] = entry_data['timestamp'].dt.hour
    exit_data['hour'] = exit_data['timestamp'].dt.hour

                                    peak_entry_time =
entry_data['hour'].value_counts().idxmax()
                                     peak_exit_time =
exit_data['hour'].value_counts().idxmax()

         print(f"Peak Entry Time: {peak_entry_time}:00 -
{peak_entry_time + 1}:00")
          print(f"Peak Exit Time: {peak_exit_time}:00 -
{peak_exit_time + 1}:00")

    entry_data['date'] = entry_data['timestamp'].dt.date
    exit_data['date'] = exit_data['timestamp'].dt.date

    entry_counts = entry_data.groupby('date').size()
    exit_counts = exit_data.groupby('date').size()
    occupancy = entry_counts - exit_counts

    print("Average Parking Occupancy by Day:")
    print(occupancy.mean())

# Usage
generate_insights(entry_data, exit_data)
```

Step 7 : Implementing the Solution in a Scalable Manner

- Tools: TensorFlow Lite, OpenVINO
- Techniques: Deploying AI models on Edge devices
- **Description** :The user-friendly interface module uses Flask to develop a web application that displays the generated insights in an accessible and interactive manner. This module provides a web interface where users can view visualizations and reports on vehicle movement patterns, parking occupancy, and vehicle matching status. The interface is designed to be intuitive and easy to navigate, allowing users to access and interpret the data effortlessly.Example Output
- **Code** :

```
# Module: app.py
from flask import Flask, render_template, request
import pandas as pd
import cv2
import pytesseract
app = Flask(__name__)
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/upload', methods=['POST'])
def upload_image():
    if 'file' not in request.files:
        return "No file part"

    file = request.files['file']
    if file.filename == '':
        return "No selected file"

    if file:
        image_path = f"static/uploads/{file.filename}"
        file.save(image_path)
        plate_number = recognize_plate(image_path)
                        update_dynamic_datasets(plate_number,
entry_dataset_path, exit_dataset_path, action='entry')
        return f"Recognized Plate: {plate_number}"

if __name__ == '__main__':
    app.run(debug=True)
```

Sample Output :

Vehicle Entry and Exit Times:

Vehicle ID: ABC123
Entry Time: 2024-07-01 08:15:23
Exit Time: 2024-07-01 17:30:45
Duration: 09:15:22

Vehicle ID: XYZ789
Entry Time: 2024-07-01 09:45:10
Exit Time: 2024-07-01 18:00:55
Duration: 08:15:45

Peak Entry Time: 08:00 - 09:00
Peak Exit Time: 17:00 - 18:00

Average Parking Occupancy by Lot:

Lot A:
  Average Occupancy: 85%
  Peak Occupancy Time: 10:00 - 12:00

Lot B:
  Average Occupancy: 70%
  Peak Occupancy Time: 14:00 - 16:00

Lot C:
  Average Occupancy: 60%
  Peak Occupancy Time: 08:00 - 09:00

Matched Vehicle License Plates:

License Plate: ABC123
Vehicle ID: ABC123
Status: Authorized

License Plate: XYZ789
Vehicle ID: XYZ789
Status: Unauthorized