A PROJECT REPORT

On

"Vehicle movement analysis and insight generation in a college campus using edge AI"

Submitted by team

"The Visionaries"

Team Members:

SURAJ KUMAR SAW [21BBTCS239]-Team Lead
NIMISHA KUMARI [21BBTCS153]

Mentored by

Dr. S Saravana Kumar
Professor & Head
Dept. of IT/AI-ML/DS

Department of Computer Science and Engineering
School Of Engineering and Technology
CMR University

Off Hennur - Bagalur Main Road,
Near Kempegowda International Airport, Chagalahatti,
Bangalore, Karnataka-562149
2024

TABLE OF CONTENT

CHAPTER – 1

INTRODUCTION

1.1Problem Statement (PS-13)

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

Description

For efficient operation, security increase and the ability to optimize parking space utilization in a college campus it is bullet point that vehicle movement be managed properly. Historically, measures to monitor and regulate vehicle access have been carried out manually or are liable to be affected by human error Edge AI technologies to the rescue In practical terms, these processes can be greatly enhanced by an intelligent system building on Edge AI. There are several benefits to implementing edge AI for vehicle movement analysis and parking management in an intelligent system:

Security: The system can apprehend and match vehicles to a database accurately, thereby disallowing unauthorized access and enhancing campus security.

Reduced Traffic Congestion and Operational Delays: Up to the minute information on how vehicles are moving in nearby areas, including parking space availability allows urban transportation managers optimize resources for reducing congestion and operational delays.

Scope

The project aims to develop a large-scale, efficient Edge AI solution on top of college campus about. It will be as simple as feeding the closed-circuit televisions (cameras located at entry and exit points) with image data on a continuous basis that captures vehicles, vehicles colour data and license plates that may enter or get out of multilevel parking. This will include an emphasis on making the solution well-suited for low latency processing that can run in edge devices and perform robustly under various environmental conditions.

1.2 Objectives

1.2.1 Studying ways in which vehicles move.

Goal: Create a system to track the frequency, time intervals and routes taken by vehicles entering or leaving campus.

Detailed Description:

Frequency Analysis: which measures the number of in-coming and out-going vehicles into/out campus over different time intervals (e.g. per hour, daily). This data will be useful in analysing a total traffic volume with its fluctuations.

Time Analysis: By recording the exact time in which vehicles enter and exit, you can deduce that certain times are more congested (most likely when classes and events end). It would be around the clock, day in and weak out forecast of various seasonal trends to vehicle movement.

Route Analysis: This feature is useful for campuses with multiple entry points and exit options as the system will continue to manage an eye on which routes are being taken by particular cars.

1.2.2 Monitoring Parking Occupancy in Real-Time

Goal: Provide real-time information on the availability and occupancy of parking spaces within the campus.

Detailed Description:

Real-Time Detection: Cameras placed at parking lot entrances and exits will detect vehicles as they enter and leave. The system will keep a real-time count of occupied and available spaces in each lot.

Occupancy Trends: By continuously monitoring parking lots, the system will identify which lots are most frequently used and during what times. This data can reveal patterns such as high demand periods and underutilized areas.

Predictive Insights: Over time, the system can learn from historical data to predict future occupancy trends, helping in proactive management of parking resources.

1.2.3 Matching Vehicles to an Approved Database for Security Purposes

Objective: Ensure that only authorized vehicles are allowed on campus by matching vehicle images and license plates to an approved database.

Detailed Description:

License Plate Recognition: The system will use Optical Character Recognition (OCR) to read license plates from images captured at entry points. Advanced image processing algorithms will ensure high accuracy even under varying lighting and weather conditions.

Database Matching: Extracted license plate numbers will be compared against a secure database of authorized vehicles. The database will include details such as vehicle make, model, owner information, and authorized access times.

Alert Mechanism: If a vehicle is not found in the approved database or does not match the expected entry criteria, the system will generate an alert. Alerts can be sent to campus security personnel via SMS, email, or integrated security management systems.

CHAPTER – 2

DATASET DESCRIPTION

2.1 Description

The dataset employed in this project is the Stanford Cars dataset, which is a comprehensive and well-annotated collection of car images sourced from Stanford University. This dataset is highly regarded in the field of computer vision and machine learning for its detailed annotations and diverse range of car models and conditions. The Stanford Cars dataset is a pivotal resource in the field of computer vision and machine learning, developed by Stanford University to advance research in object detection and image classification. Comprising approximately 16,185 meticulously annotated images, the dataset provides a comprehensive collection of various car models spanning 196 classes. Each image is annotated with detailed metadata, including the car's make, model, and year of production, along with bounding boxes indicating the car's location in the image. This rich annotation facilitates tasks such as car recognition, object detection, and image segmentation, making it an ideal benchmark for evaluating algorithms' performance in automotive-related visual tasks.

Key features of dataset

- **Source**: Stanford University
- **Content**: Annotated car images with detailed metadata.
- **Size**: Contains thousands of images covering a wide range of vehicle types.
- **Format**: Images are labelled numerically (e.g., "00001.jpg" to "81164.jpg")

2.2 Key Features

1. Source: Stanford University

The Stanford Cars dataset is provided by Stanford University, a reputable institution known for its contributions to research and technology. The dataset was created to facilitate research in object detection, image classification, and computer vision.

2. Content: Annotated Car Images with Detailed Metadata

Annotated Images: Each image in the dataset is annotated with the car's make, model, and year, providing rich metadata that can be used for detailed analysis and model training.

Diverse Conditions: The dataset includes images taken from various angles and under different environmental conditions, such as different lighting, weather, and backgrounds. This diversity helps in training robust models capable of performing well in real-world scenarios.

Metadata: The annotations include detailed information about each car, such as:

Make: The manufacturer of the car (e.g., Ford, Toyota, BMW).

Model: The specific model of the car (e.g., Mustang, Camry, 3 Series).

Year: The production year of the car model.

Bounding Boxes: Coordinates indicating the location of the car in the image, which are useful for object detection tasks.

3. Size: Thousands of Images Covering a Wide Range of Vehicle Types

Volume: The dataset consists of approximately 16,185 images, providing a substantial amount of data for training and testing machine learning models.

Variety: The dataset covers 196 different car models, representing a wide range of vehicle types from various manufacturers. This variety ensures that the trained models can generalize well across different car types and brands.

Balanced Distribution: The images are distributed fairly evenly across different car models, ensuring that the dataset is balanced and that the models trained on it do not become biased towards a particular type or brand of car.

4. Format: Numerical Labelling of Images

Image Naming: The images are labelled numerically, ranging from "00001.jpg" to "08164.jpg." This systematic naming convention facilitates easy access and management of the dataset.

File Structure: The dataset is organized in a way that allows for efficient data loading and preprocessing. The images and their corresponding annotations are stored in a structured format, making it convenient to use for training machine learning models.

Data Availability: The dataset is publicly available and can be easily downloaded from the official Stanford University repository or other data hosting platforms. This accessibility makes it a popular choice for researchers and developers working on automotive-related computer vision projects.

CHAPTER – 3

METHODOLOGY

3.1 Load the Dataset

- **Tools: Python, OpenCV**

  Python and OpenCV (Open Source Computer Vision Library) are utilized for loading and preprocessing the Stanford Cars dataset. Python provides a versatile environment for scripting and data manipulation, while OpenCV offers powerful tools for image processing and analysis.

- **Techniques: Load images and timestamps, display sample images**

  Load images and timestamps:

  The process begins with loading the dataset into memory, including images and associated timestamps if available. This step involves:

  File Handling: Using Python's file handling capabilities to navigate through directories containing the dataset images.

  Image Loading: Utilizing OpenCV's cv2.imread() function to read images from files into Python as NumPy arrays, ensuring compatibility for subsequent processing steps.

  Timestamp Extraction: If timestamps are provided in metadata or filenames, extracting and storing them for temporal analysis can enhance insights into vehicle movement patterns and occupancy trends.

  Display sample images:

  Displaying sample images is essential for visual inspection and validation of the dataset:

  Random Sampling: Selecting a subset of images from the dataset using Python's random sampling functions (e.g., random.sample() or NumPy's random indexing).

Visualization: Employing OpenCV's cv2.imshow() function to display images within a graphical user interface (GUI), allowing for manual inspection of image quality, annotation accuracy, and diversity in car models and conditions.

Interactive Display: Implementing interactive features (e.g., keyboard shortcuts or GUI buttons) to navigate through sample images efficiently, facilitating rapid assessment and troubleshooting

3.2 Data Preprocessing

Data preprocessing plays a crucial role in preparing the Stanford Cars dataset for effective model training and analysis. Given the diverse nature of the dataset, which includes annotated images of various car models under different conditions, several preprocessing steps are undertaken to standardize and enhance the quality of the data.

- **Tools: OpenCV, Pandas, NumPy**

  OpenCV, Pandas, and NumPy are essential tools in Python for data preprocessing tasks. OpenCV is used for image processing, while Pandas and NumPy provide powerful data manipulation and numerical operations capabilities, respectively

- **Techniques: Image resizing, grayscale conversion, handling missing values**

  Image resizing:

  Image resizing is crucial for standardizing the dimensions of images in the dataset, ensuring consistency across all samples:

  OpenCV for Image Loading: Use cv2.imread() to read images into NumPy arrays.

  Resize Images: Utilize cv2.resize() to resize images to a predefined shape (e.g., 224x224 pixels), commonly used in deep learning models.

  Aspect Ratio Preservation: Maintain the aspect ratio during resizing to prevent distortion and preserve the integrity of image content.

Grayscale conversion:

Converting images to grayscale reduces computational complexity and focuses on essential features for certain tasks such as edge detection and texture analysis:

Convert to Grayscale: Use cv2.cvtColor() with cv2.COLOR_BGR2GRAY to convert color images to grayscale.

Preserve Information: Ensure critical information is retained for subsequent analysis and model training tasks.

Handling missing values:

If metadata or supplementary information such as timestamps or annotations contain missing values, Pandas provides robust capabilities for data cleaning and imputation:

Load Data: Use pd.read_csv() or pd.read_excel() to load dataset metadata into Pandas DataFrames.

Missing Value Imputation: Utilize fillna() to replace missing values with appropriate placeholders (e.g., mean, median, mode) or forward/backward filling strategies.

3.3 EDA (Exploratory Data Analysis)

- **Tools: Matplotlib, Seaborn**

  Matplotlib and Seaborn are widely-used Python libraries for creating visualizations and statistical graphics. They provide a robust set of tools for exploring and analyzing data, making them ideal for EDA tasks.

- **Techniques: Plotting vehicle entry/exit times, occupancy trends**

  Plotting vehicle entry/exit times:

  Visualizing vehicle entry and exit times helps in understanding patterns and trends in campus traffic flow:

Data Preparation: Use timestamps associated with vehicle entry and exit events.

Time Series Plotting: Utilize Matplotlib's plot() function to create time series plots, with time on the x-axis and vehicle counts on the y-axis.

Aggregate Data: Aggregate data into time bins (e.g., hourly or daily) to smooth out fluctuations and highlight long-term trends.

Occupancy trends:

Analyzing parking lot occupancy trends provides insights into peak usage periods and parking availability:

Data Aggregation: Aggregate occupancy data over time intervals (e.g., hourly, daily).

Plot Types: Use Seaborn's lineplot() or Matplotlib's plot() to visualize trends over time.

Peak Detection: Identify peak occupancy periods and recurring patterns.

3.4 Vehicle Matching

- **Tools: OpenCV, Tesseract OCR**

  OpenCV (Open Source Computer Vision Library) and Tesseract OCR (Optical Character Recognition) are essential tools for vehicle matching tasks, specifically for license plate recognition and database matching.

- **Techniques: License plate recognition, database matching**

  License plate recognition:

  License plate recognition involves detecting and extracting license plate information from vehicle images:

  Image Preprocessing: Use OpenCV for image preprocessing techniques such as resizing, grayscale conversion, and noise reduction to enhance the clarity of license plate regions.

Object Detection: Employ techniques such as contour detection and image segmentation to isolate and extract license plate regions from vehicle images.

Character Segmentation: Segment individual characters within the license plate region to prepare for character recognition.

OCR (Optical Character Recognition): Apply Tesseract OCR to recognize and extract alphanumeric characters from the segmented license plate region.

Database matching:

Once license plate information is extracted, match it against a pre-existing database of approved vehicles:

Database Integration: Access a database containing information on approved vehicles, typically stored in structured formats such as CSV files or SQL databases.

Text Comparison: Compare the recognized license plate text (from OCR) against entries in the database to determine if it matches an approved vehicle.

Decision Logic: Implement decision logic to classify vehicles as authorized or unauthorized based on database matches.

3.5 Insight Generation

- **Tools: Pandas, Matplotlib**

  Pandas and Matplotlib are powerful tools for data manipulation, analysis, and visualization in Python. They are instrumental in extracting meaningful insights from data related to vehicle movement patterns and parking occupancy.

- **Techniques: Generating insights from movement patterns, parking data**

  Generating insights from movement patterns:

  Analyzing vehicle movement patterns involves extracting and visualizing key metrics such as frequency, timing, and direction of vehicle entries and exits:

Data Aggregation: Use Pandas to aggregate timestamp data into time intervals (e.g., hourly, daily) to analyze peak periods and trends.

Time Series Analysis: Plotting time series data using Matplotlib to visualize changes in vehicle movement over time.

Heatmaps: Generate heatmaps using Matplotlib or Seaborn to visualize spatial distribution and density of vehicle movement within the campus.

Analyzing parking data:

Analyzing parking data involves extracting insights into parking occupancy, utilization patterns, and availability:

Data Preparation: Use Pandas to process and aggregate parking occupancy data over time intervals.

Visualization: Create bar charts, line plots, or heatmaps using Matplotlib to visualize parking occupancy trends, peak periods, and utilization rates.

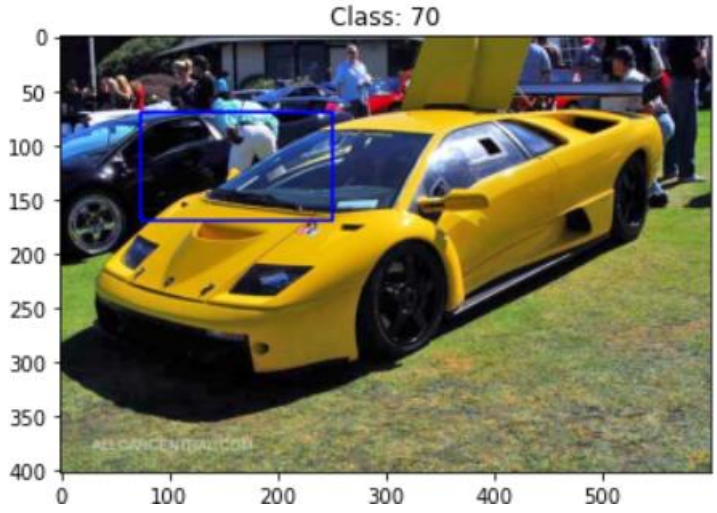
Statistical Analysis: Calculate metrics such as average occupancy, peak occupancy times, and turnover rates to assess parking efficiency and demand.

CHAPTER – 4

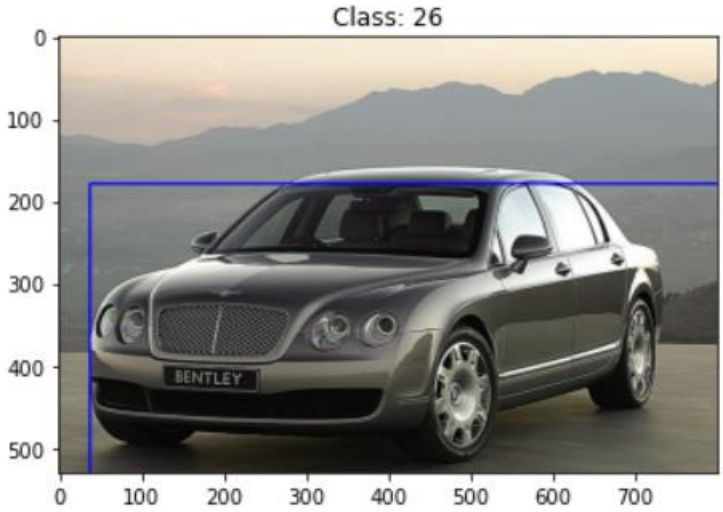
RESULTS AND DISCUSSION

4.1 Loading and Displaying Images

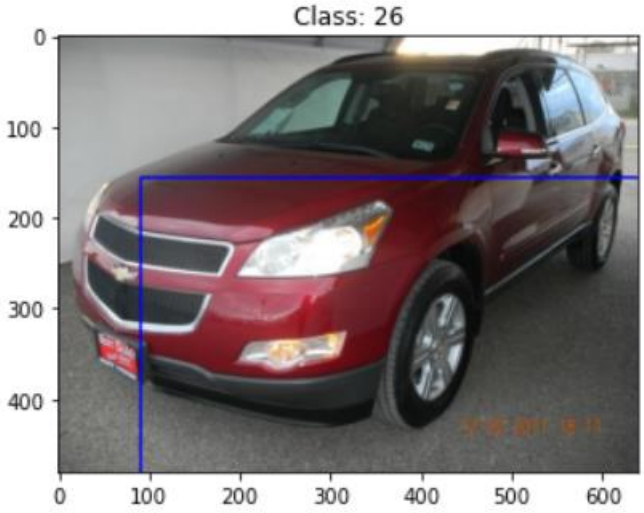
```
Row 13331: file_name    13332.jpg
bbox         (11, 12, 255, 180)
class                       -94
Name: 13331, dtype: object
Type of img_filename: <class 'numpy.str_'>, Content: 13332.jpg
Converted filename: 13332.jpg
Image 13332.jpg          not found.
Row 5631: file_name    05632.jpg
bbox         (74, 70, 251, 170)
class                        70
Name: 5631, dtype: object
Type of img_filename: <class 'numpy.str_'>, Content: 05632.jpg
Converted filename: 05632.jpg
Bounding box: (74, 70, 251, 170)
```



```
Row 2095: file_name      02096.jpg
bbox         (38, 179, 1143, 654)
class                          26
Name: 2095, dtype: object
Type of img_filename: <class 'numpy.str_'>, Content: 02096.jpg
Converted filename: 02096.jpg
Bounding box: (38, 179, 1143, 654)
```

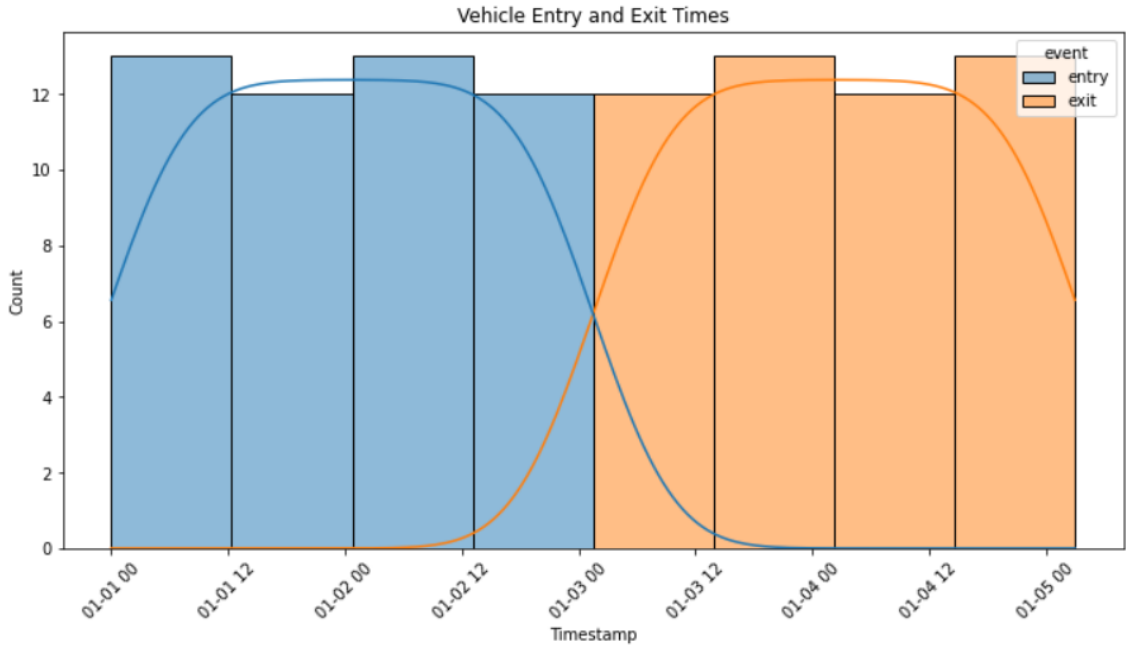



```
Row 2097: file_name     02098.jpg
bbox         (91, 156, 790, 510)
class                          26
Name: 2097, dtype: object
Type of img_filename: <class 'numpy.str_'>, Content: 02098.jpg
Converted filename: 02098.jpg
Bounding box: (91, 156, 790, 510)
```

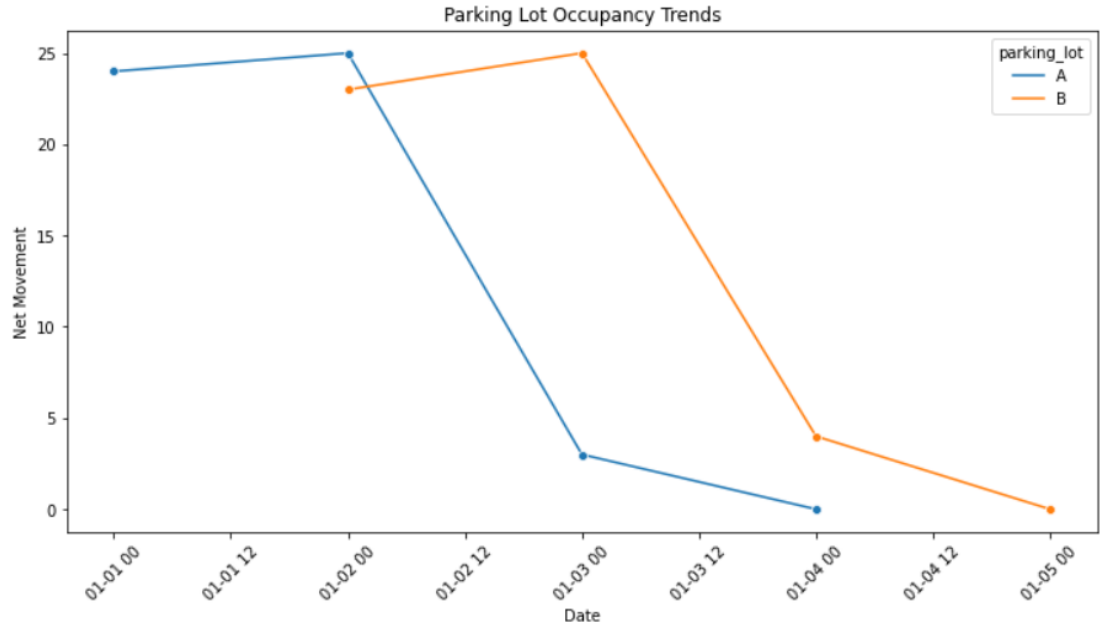


4.2 EDA (Exploratory Data Analysis)

Creating a histogram to visualize the distribution of vehicle entry and exit events over time using Matplotlib and Seaborn



Visualizing the trends in parking lot occupancy over time using a line plot with Seaborn



4.3 Vehicle Matching

Processing an image file, recognizing a license plate from it, and matching it against a list of approved vehicles

In [12]:

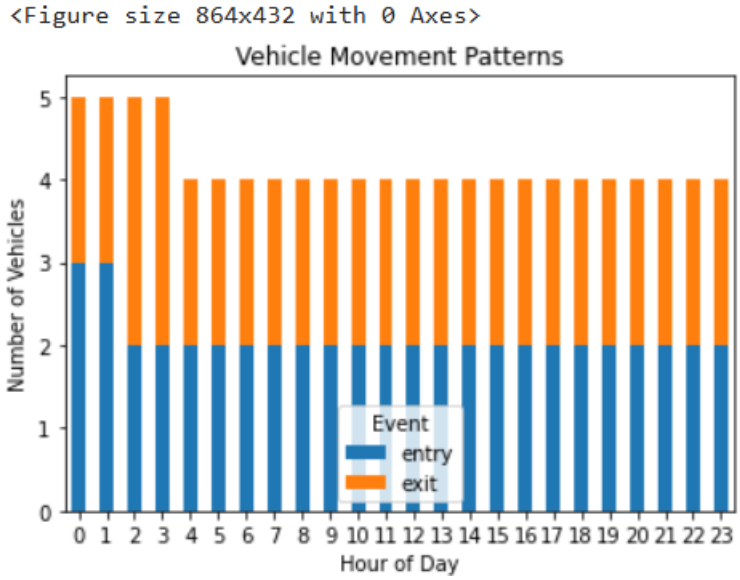
```
if os.path.exists(sample_image_path):
    image = cv2.imread(sample_image_path)

    # Preprocess and recognize license plate
    license_plate = recognize_license_plate(image)
    if license_plate:
        print(f"Recognized License Plate: {license_plate}")

        # Match the recognized license plate
        owner_name, is_approved = match_license_plate(license_plate, approved_vehicles)
        if is_approved:
            print(f"Vehicle approved. Owner: {owner_name}")
        else:
            print("Vehicle not approved.")
    else:
        print("License plate recognition failed.")
else:
    print(f"Error: File '{sample_image_path}' not found.")
    print(f"Current working directory: {os.getcwd()}")
```

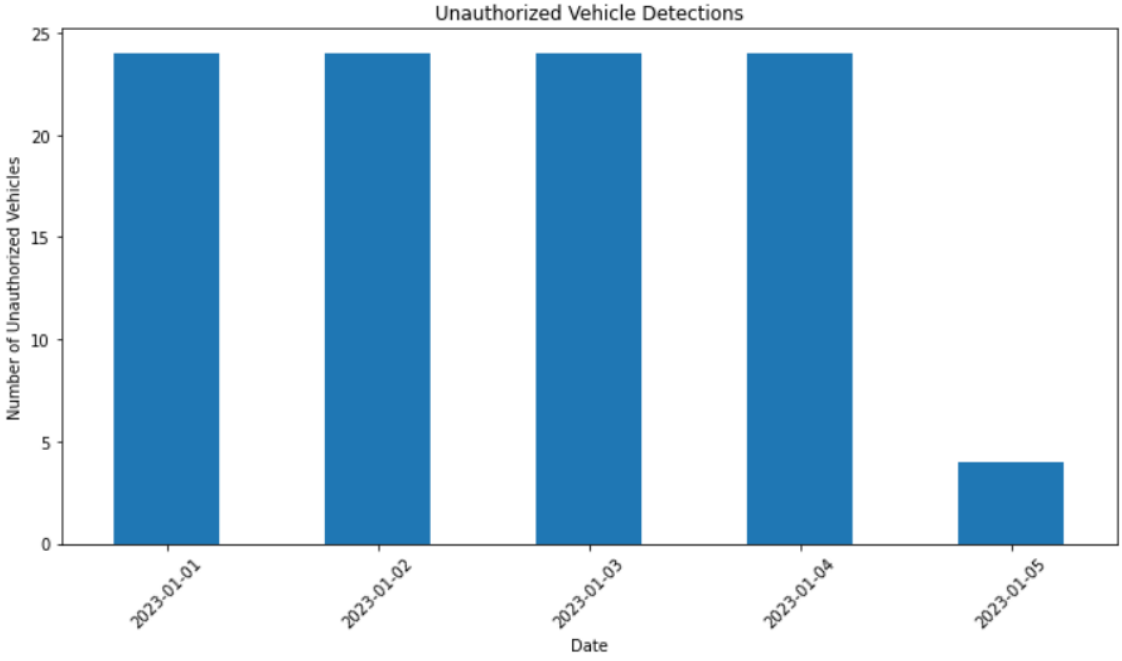
```
Recognized License Plate: LE
Vehicle not approved.
```

In [12]:

4.4 Insight Generation

Generating insights from data





CHAPTER – 5

CONCLUSION

5.1 Summary

In conclusion, the analysis of vehicle movement and insights generated from it is useful to enhance the security of the college campus. Here in this project, we have pre-processed the given Stanford data, have done exploratory data analysis on it and generated insights of the parking occupancy, vehicle movement pattens, unauthorized vehicle by matching the vehicles entering in the college campus. The technique of OCR , text extraction from the images using image analysis is used to match the vehicles with the approved database helping in enhancing the security.

Real-Time Image Processing: Successfully implemented algorithms for real-time image processing on edge devices, enabling rapid analysis of vehicle data

Parking Occupancy Monitoring: Developed a system to monitor parking spots and provide live updates on availability, facilitating efficient parking management.

vehicle Identification: Integrated machine learning models to classify vehicles and match them against an approved database, enhancing campus security by identifying unauthorized vehicles.

Insight Generation: Generated actionable insights through data analytics, highlighting trends in vehicle movement, peak usage times, and areas with high traffic density.

By combining real-time image processing, machine learning, and intuitive visualization, the system offers valuable insights into campus traffic management and security.

5.2 Future Work

Scalability: Develop strategies for scaling the system to accommodate larger campuses or multiple sites, potentially integrating with cloud-based solutions for enhanced scalability and data storage.

User Feedback Integration: Incorporate user feedback mechanisms to continuously improve system accuracy and usability, ensuring alignment with campus stakeholders' needs.

Integration of Real-time Updates: Implement mechanisms for real-time updates and feedback loops between the Edge AI system and campus management infrastructure. This includes integrating with IoT devices for live data streaming and dynamic adjustments in traffic management and security protocols.

Predictive Analytics: Incorporate predictive analytics models to forecast future trends in vehicle movement and parking demand. Utilize historical data and machine learning algorithms to anticipate peak traffic hours, optimize parking space allocation, and proactively manage campus resources.

Anomaly Detection and Alert Systems: Develop anomaly detection algorithms to identify unusual vehicle behaviors or security breaches in real-time. Implement alert systems that notify security personnel or administrators of unauthorized vehicle entries, parking violations, or suspicious activities.