

DEEP LEARNING

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

Abhay Prasad, Anaswara Biju , and Aswathy Satheesh

Saintgits Group of Institutions, Kottayam, Kerala

Abstract: "Vehicle Movement Analysis and Insight Generation in a College Campus using Edge AI" explores the application of edge artificial intelligence (AI) for analyzing and generating insights from vehicle movement within a college campus. This study focuses on leveraging edge computing to process real-time data from campus surveillance systems, aiming to enhance security, optimize traffic flow, and improve operational efficiency within the campus environment. The abstract highlights the integration of AI algorithms at the network edge to enable rapid data processing and decision-making, ultimately contributing to a safer and more efficient campus infrastructure.

Keywords: Edge AI, vehicle movement analysis, campus security, traffic optimization, operational efficiency, real-time data processing

1 Introduction

In recent years, the rapid advancement of Artificial Intelligence (AI) and its deployment on Edge devices have significantly transformed various domains, including transportation, security, and facility management. A crucial application area involves the analysis of vehicle movement and parking management within controlled environments, such as college campuses. Efficient monitoring and management of vehicle movements in such areas are critical for ensuring security, optimizing parking resources, and maintaining smooth traffic flow. In a detailed study, the proposed computer vision-based vehicle detection system leveraging deep learning techniques has demonstrated a vehicle detection accuracy of 95% and an OCR accuracy of 90% for license plate recognition. By providing insights into vehicle movement patterns, parking occupancy, and matching vehicles to an approved database, this system empowers campus administrators with real-time data-driven

decision-making capabilities. The result is enhanced security measures, optimized parking resources, and improved overall operational efficiency. By offering a comprehensive intelligent transportation system solution, this project aims to significantly improve traffic management and security within a college campus environment. The modular architecture of the system ensures scalability and customization, allowing adaptation to various hardware configurations and surveillance scenarios. Extensive experiments conducted using real-world scenarios and benchmark datasets have confirmed the system's high efficiency and accuracy in vehicle detection and license plate recognition tasks.

2 Dataset Description

The dataset for this project was exclusively gathered from the college campus environment to accurately reflect real-world vehicle movement scenarios. A total of 50 images were captured, encompassing various locations, including entry points and parking lots. Following thorough analysis and image enhancement, 12 images were allocated for training and testing the vehicle detection model using YOLOv8, ensuring robust performance across diverse conditions. Additionally, 15 images were selected for training and testing the OCR model for license plate recognition, focusing on different plate designs and lighting conditions. From these, another 5 images were collected, and 2 were chosen for analyzing parking lot occupancy and management, capturing varying occupancy levels throughout different times of the day. This approach ensured that the dataset comprehensively covered all necessary aspects for developing an accurate solution tailored to the specific needs of the college campus.

3 Libraries Used

In the project for various tasks, following packages are used.

```
Pandas  
cv2  
os  
Matplotlib.pyplot  
easyocr  
ultralytics
```

4 Methodology

Data Collection: A dataset comprising vehicle images was meticulously collected exclusively from various locations within the college campus. Utilizing existing sources for image acquisition circumvented the need for additional camera installations. The collected images were stored efficiently using cloud or edge storage solutions to facilitate subsequent processing.

Data Preprocessing: Image data underwent meticulous preprocessing to ensure optimal quality for analysis. Techniques including noise reduction and contrast adjustment were systematically applied to enhance image clarity and detail. Careful selection

of relevant images from the dataset further refined the dataset for subsequent model training and analysis.

Object Detection: YOLOv8, known for its efficiency in real-time object detection, was selected as the core algorithm for identifying vehicles within campus images. The model was pretrained using a labeled dataset of vehicle images and rigorously tested to achieve a targeted accuracy threshold of 85%.

License Plate Recognition: Optical Character Recognition (OCR) techniques, integrated with EasyOCR, were employed to extract and interpret license plate information from detected vehicles. This integration with YOLOv8 allowed for accurate isolation and recognition of license plate text, ensuring comprehensive vehicle identification capabilities.

Data Analysis and Insight Generation: Recognized license plates were cross-referenced with an authorized vehicle database to distinguish between authorized and unauthorized vehicles on campus. Additionally, analysis of the collected images enabled precise determination of parking occupancy rates and available spaces, facilitating efficient parking lot management. Patterns in vehicle movement, such as peak hours, frequent visitors, and unusual activities, were identified through thorough analysis of detected vehicles and recognized license plates.

5 Tools Used

Python: Python served as the primary programming language for implementing algorithms and conducting data analysis.

Google Colab: Google Colab provided a collaborative and powerful development environment, leveraging cloud computing capabilities for model development, training, and analysis.

YOLOv8: You Only Look Once version 8, is a state-of-the-art object detection framework designed for fast and accurate vehicle identification and tracking in images. It excels in detecting multiple objects simultaneously within a single frame, making it ideal for real-time applications where speed and precision are critical.

EasyOCR: EasyOCR is an Optical Character Recognition (OCR) tool widely utilized for reading and recognizing license plate numbers from detected vehicles. It boasts high accuracy in text recognition across different languages and fonts, making it versatile for applications requiring robust and reliable license plate identification capabilities.

6 Implementation

A Python-based application designed to analyze vehicle movement and provide insights within a college campus. The system detects vehicles, recognizes license plates, and analyzes movement patterns to offer valuable insights on parking occupancy and vehicle authorization.

Key Components:

Vehicle Detection Module: YOLOv8: For high-accuracy vehicle detection

License Plate Recognition Module: OCR Engine: Extracts license plate information from detected vehicles.

Data Aggregation and Analysis Engine: Collects and analyzes vehicle data (timestamps, license plates).

input/Output Mechanism: input: Images collected from college campus Authorized vehicle database.

Output: vehicle detection , license plate recognition, Total, authorized, and unauthorized vehicle count, Parking lot occupancy rates

7 Results & Discussion

The project "Vehicle Movement Analysis and Insight Generation in a College Campus using Edge AI" successfully implemented an Edge AI-based solution to analyze vehicle movement patterns, parking occupancy, and match vehicles to an approved database. Using image data from campus locations, the system accurately monitored traffic flows, optimized parking utilization by providing real-time updates on parking availability, and enhanced campus security through effective license plate recognition. The insights gained enable data-driven decisions for campus management, supporting proactive traffic management strategies and improving overall operational efficiency. The project underscores the efficacy of Edge AI in transforming campus management practices, laying the groundwork for future enhancements in smart campus technologies and urban mobility solutions.

float subcaption

Serial Number	Class	License Plate	Authorized
1	car	KLOBBK LLL6 LOBBK LLLG]	Unauthorized
2	car	KLo54J61351 [Ko5ajgi35	Authorized
3	car	KIOSba 407 'KL05BA7407'	Authorized
4	car	Klolageee5 Klolagebe5	Authorized
5	car	KLOSAuO130 'KL05Au0130	Authorized
6	car	Klo5as8628 KLosase628	Authorized
7	car		Unauthorized
8	car	IKLOSAY5959 KLO5AY5959	Authorized
9	car	LOZCP5478 KLO7CP5478]	Authorized
10	car	KLOABY7325 KLOABV7325/	Unauthorized
11	car	KL3603239][KL3603339	Authorized
12	car	KLo7bk968] KLO7bk968]	Authorized
13	car	J04L054y59e0 Liloshussee]	Unauthorized
14	car	KL 29 E 6u80] (KL. 29 E.6380.	Authorized
15	car	KL2702203 KL27 02303	Authorized

(a) Csv File Showing Authorized or Unauthorized.

allotted_vehicles_to_the_slot.csv X		
1 to 3 of 3 entries Filter		
Vacant Spaces	Allotted Vehicles	Class
3	KL2702203 KL27 02303	Assigned
2	KLo54J61351 [Ko5ajgi35	Assigned
1	KL3603239][KL3603339	Assigned
Show 10 per page		

(b) Alloted Vehicles to the parking slot.

Figure 1: Csv Files

As shown in Figure 1: a) Out of 15 vehicles, 12 vehicles are authorized. Any other vehicle that enters is considered unauthorized. If the total number of vehicles changes, only those 12 vehicles will be considered authorized, and the remaining ones will be unauthorized. b) According to our dataset, the maximum occupancy of the parking lot is 10. Seven

vehicles occupied the parking slots in our dataset, so the remaining slots that could be occupied were three. Therefore, from the authorized vehicle list, three randomly selected vehicles were assigned to the three vacant parking slots.

8 Conclusions

The project "Vehicle Movement Analysis and Insight Generation in a College Campus Using Edge AI" successfully implemented a solution to analyze vehicle movement patterns, parking occupancy, and match vehicles to an approved database. By leveraging images capturing vehicle photos and license plates from various campus locations, the system effectively monitored traffic flows, optimized parking space utilization, and enhanced campus security by identifying authorized vehicles. The solution also provided detailed parking allotment information, including real-time updates on parking availability and utilization trends across different areas of the campus.

Acknowledgments

We would like to express our heartfelt gratitude and appreciation to Intel® Corporation for providing an opportunity to this project. First and foremost, we would like to extend our sincere thanks to our team mentor Dr. Pradeep C for his invaluable guidance and constant support throughout the project. We are deeply indebted to our college Saintgits College of Engineering and Technology for providing us with the necessary resources, and sessions on machine learning. We extend our gratitude to all the researchers, scholars, and experts in the field of machine learning and natural language processing and artificial intelligence, whose seminal work has paved the way for our project. We acknowledge the mentors, institutional heads, and industrial mentors for their invaluable guidance and support in completing this industrial training under Intel® -Unnati Programme whose expertise and encouragement have been instrumental in shaping our work. []

References

- [1] AMATO, G., CARRARA, F., FALCHI, F., GENNARO, C., MEGHINI, C., AND VAIRO, C. Deep learning for decentralized parking lot occupancy detection. *Expert Systems with Applications* 72 (2017), 327–334. <https://doi.org/10.1016/j.eswa.2016.10.055>.
- [2] GONTHINA, N., KATKAM, S., POLA, R. A., PUSULURI, R. T., AND PRASAD, L. V. N. Parking slot detection using yolov8. In *2023 3rd International Conference on Mobile Networks and Wireless Communications (ICMNWC)* (Tumkur, India, 2023), pp. 1–5. 10.1109/ICMNWC60182.2023.10435799.
- [3] IRIAWAN, A. D., AND SUNYOTO, A. Automatic license plate recognition system in indonesia using yolov8 and easyocr algorithm. In *2023 6th International Conference on Information and Communications Technology (ICOIACT)* (2023), pp. 384–388. 10.1109/ICOIACT59844.2023.10455908.

- [4] KHAN, S., VOHRA, S., SIDDIQUE, S. A., ABRO, A. A., AND EBRAHIM, M. A computer vision-based vehicle detection system leveraging deep learning. In *2024 IEEE 1st Karachi Section Humanitarian Technology Conference (KHI-HTC)* (2024), pp. 1–7. 10.1109/KHI-HTC60760.2024.10482163.
- [5] MOUSSAOUI, H., AKKAD, N. E., BENSLIMANE, M., EL-SHAFI, W., BAIHAN, A., HEWAGE, C., AND RATHORE, R. S. Enhancing automated vehicle identification by integrating yolo v8 and ocr techniques for high-precision license plate detection and recognition. *Scientific Reports* 14 (Jun. 2024), Art. no. 14389. 10.1038/s41598-024-71234-9.
- [6] PATIL, S. S., PATIL, S. H., PAWAR, A. M., BEWOOR, M. S., KADAM, A. K., PATKAR, U. C., WADARE, K., AND SHARMA, S. Vehicle number plate detection using yolov8 and easyocr. In *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (2023), pp. 1–4. 10.1109/ICCCNT56998.2023.10307420.
- [7] SINGH, P., SURYAWANSHI, M. S., AND TAK, D. Smart fleet management system using iot, computer vision, cloud computing and machine learning technologies. In *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)* (2019), pp. 1–8. 10.1109/I2CT45611.2019.9033578.

A Main code sections for the solution

A.1 Cloning the Repository

```
!git clone https://github.com/itsmeabhay01/VICTUS.git
cd /content/VICTUS
```

A.2 Running the Prediction Script with OCR on Test Images

```
!python predictWithOCR.py model='/content/VICTUS/best.pt' source='/content/VICTUS/
data/testimages'
```

A.3 Display Prediction Results

```
results_dir = '/content/VICTUS/runs/detect/train'
image_files = [os.path.join(results_dir, img) for img in os.listdir(results_dir)
                if img.endswith('.jpg')]

# Loop through the images and plot them
for idx, image_file in enumerate(image_files):
    row = idx // num_columns
    col = idx % num_columns
    ax = axes[row, col] if num_rows > 1 else axes[col]
    image = Image.open(image_file)
    ax.imshow(image)
    ax.axis('off') # Hide the axis
# Hide any remaining empty subplots
for idx in range(len(image_files), num_rows * num_columns):
```



```

row = idx // num_columns
col = idx % num_columns
ax = axes[row, col] if num_rows > 1 else axes[col]
ax.axis('off') # Hide the axis
plt.tight_layout()
plt.show()

```

A.4 Vehicle Detection with YOLOv8 and Image Display

```

def display_images(images, cols=3):
    rows = len(images) // cols + int(len(images) % cols != 0)
    fig, axes = plt.subplots(rows, cols, figsize=(20, rows * 7))
    for i, img in enumerate(images):
        row = i // cols
        col = i % cols
        axes[row, col].imshow(img)
        axes[row, col].axis('off')
    # Detect vehicles in each image
    all_images = []
    for img_file in os.listdir(image_folder):
        img_path = os.path.join(image_folder, img_file)
        if img_file.lower().endswith(('png', '.jpg', '.jpeg')): # Ensure we only
                                                                process image files
            print(f"Processing image: {img_file}")
            results = model(img_path)
            # Print the results for cars
            for car_result in car_results:
                print(f"Class: {car_result[0]}, Confidence: {car_result[1]}, BBox: {
                    car_result[2]}")

            # Append the annotated image to the list
            all_images.append(img_rgb)
    # Display all annotated images in a grid
    display_images(all_images, cols=3)

```

A.5 Converting to Csv File

```

def recognize_license_plate(image, bbox):
    x1, y1, x2, y2 = map(int, bbox)
    plate_image = image[y1:y2, x1:x2]
    results = reader.readtext(plate_image)
    license_plate = " ".join([res[1] for res in results]) if results else ""
    return license_plate

def display_images(images, cols=3):
    rows = len(images) // cols + int(len(images) % cols != 0)
    fig, axes = plt.subplots(rows, cols, figsize=(20, rows * 7))
    for i, img in enumerate(images):
        row = i // cols
        col = i % cols
        axes[row, col].imshow(img)
        axes[row, col].axis('off')
    # Detect vehicles and license plates
    results_list = []
    all_images = []
    for img_file in os.listdir(image_folder):
        img_path = os.path.join(image_folder, img_file)

```

```

# Print the results for cars
for car_result in car_results:
    print(f"Class: {car_result[0]}, BBox: {car_result[1]}")
# Recognize license plates for detected cars
for car_result in car_results:
    bbox = car_result[1]
    license_plate = recognize_license_plate(img, bbox)
    entry_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    results_list.append([entry_time, img_file, car_result[0], license_plate])
# Append the annotated image to the list
all_images.append(img_rgb)
# Save results to CSV
df = pd.DataFrame(results_list, columns=['Entry Time', 'Image File', 'Class', '
                                         License Plate'])
df.to_csv(results_csv, index=False)

```

A.6 Authorizing the Detected vehicles in the Campus

```

def recognize_license_plate(image, bbox):
    x1, y1, x2, y2 = map(int, bbox)
    plate_image = image[y1:y2, x1:x2]
    results = reader.readtext(plate_image)
    license_plate = " ".join([res[1] for res in results]) if results else ""
    return license_plate

def display_images(images, cols=3):
    rows = len(images) // cols + int(len(images) % cols != 0)
    fig, axes = plt.subplots(rows, cols, figsize=(20, rows * 7))
    for i, img in enumerate(images):
        row = i // cols
        col = i % cols
        axes[row, col].imshow(img)
        axes[row, col].axis('off')
    # Detect vehicles and license plates
    results_list = []
    all_images = []
    # Print the results for cars
    for car_result in car_results:
        print(f"Class: {car_result[0]}, BBox: {car_result[1]}")
    # Save results to CSV
    df = pd.DataFrame(results_list, columns=['Entry Time', 'Image File', 'Class', '
                                         License Plate'])
    df.to_csv(results_csv, index=False)
    # Create a new DataFrame with serial number and authorization status
    df['Serial Number'] = range(1, len(df) + 1)
    df['Authorized'] = 'Yes' # Assuming all detected vehicles are authorized
    # Save the new DataFrame to a new CSV file
    df[['Serial Number', 'Class', 'License Plate', 'Authorized']].to_csv(
        authorized_csv, index=False)

```

A.7 Vehicle Matching(To detect whether the Vehicle is Authorized or Unauthorized)

```

!python predictWithOCR.py model='/content/VICTUS/best.pt' source='/content/VICTUS/
data/completeimages'

```



```

def recognize_license_plate(image, bbox):
    x1, y1, x2, y2 = map(int, bbox)
    plate_image = image[y1:y2, x1:x2]
    results = reader.readtext(plate_image)
    license_plate = " ".join([res[1] for res in results]) if results else ""
    return license_plate

def display_images(images, cols=3):
    rows = len(images) // cols + int(len(images) % cols != 0)
    fig, axes = plt.subplots(rows, cols, figsize=(20, rows * 7))
    for i, img in enumerate(images):
        row = i // cols
        col = i % cols
        axes[row, col].imshow(img)
        axes[row, col].axis('off')
        # Recognize license plates for detected cars
        for car_result in car_results:
            bbox = car_result[1]
            license_plate = recognize_license_plate(img, bbox)
            entry_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
            is_authorized = 'Authorized' if license_plate in authorized_license_plates else 'Unauthorized'
            results_list.append([entry_time, img_file, car_result[0], license_plate, is_authorized])

    # Append the annotated image to the list
    all_images.append(img_rgb)
    # Create a new DataFrame with serial number and authorization status
    vehicle_entry_df = pd.DataFrame(results_list, columns=['Entry Time', 'Image File', 'Class', 'License Plate', 'Authorized'])
    vehicle_entry_df['Serial Number'] = range(1, len(vehicle_entry_df) + 1)
    # Save the new DataFrame to a new CSV file
    vehicle_entry_df[['Serial Number', 'Class', 'License Plate', 'Authorized']].to_csv(
        vehicle_entry_csv_path, index=False)

```

A.8 Parking Lot Occupancy

```

# Define the path to the parking lot image
parking_lot_image_path = '/content/VICTUS/data/parkinglotimage.jpg'
def display_image_with_bboxes(image, bboxes, labels):
    for bbox, label in zip(bboxes, labels):
        start_point = (int(bbox[0]), int(bbox[1]))
        end_point = (int(bbox[2]), int(bbox[3]))
        color = (0, 255, 0) # Green color for bounding box
        thickness = 2
        image = cv2.rectangle(image, start_point, end_point, color, thickness)
    # Display the annotated image with bounding boxes
    annotated_img = display_image_with_bboxes(img_rgb, bboxes, labels)
    plt.figure(figsize=(10, 10))
    plt.imshow(annotated_img)
    plt.axis('off')
    plt.show()
    print(f"Number of vehicles detected: {vehicle_count}")
    # Calculate vacant slots
    vacant_slots = TOTAL_PARKING_SLOTS - vehicle_count
    # Create a DataFrame
    data = {
        'Parking Slots': [TOTAL_PARKING_SLOTS],
        'Occupied': [vehicle_count],

```

```
'Vacant': [vacant_slots]
}
df = pd.DataFrame(data)
# Define the output CSV path
output_csv_path = '/content/parking_slots.csv'
# Save to CSV
df.to_csv(output_csv_path, index=False)
print(f"Parking slot details saved to {output_csv_path}")
```

A.9 Assigning Vehicles to Parking Lot

```
# Load the parking lot status CSV
parking_lot_status = pd.read_csv('/content/parking_slots.csv')
# Check the number of vacant spaces (assuming the correct column name is 'Vacant')
total_vacant_spaces = parking_lot_status['Vacant'].sum()
# Check if there are any vacant spaces
if total_vacant_spaces == 0:
    print("No vacant parking spaces available.")
else:
    # Randomly select the number of vehicles to be assigned based on the number of
    # vacant spaces
    num_authorized_vehicles = len(authorized_vehicles)
    # Randomly select the vehicles to be assigned
    num_to_assign = min(total_vacant_spaces, num_authorized_vehicles)
    assigned_vehicles = authorized_vehicles.sample(n=num_to_assign)
    # Filter and create a DataFrame for the vacant spaces to be assigned
    vacant_spaces = parking_lot_status.loc[parking_lot_status['Vacant'] > 0].copy()
    # Save the results to a new CSV file
    output_csv = '/content/allotted_vehicles_to_the_slot.csv'
    assigned_parking.to_csv(output_csv, index=False)
    print(f"Allotted vehicles to the slot have been saved to {output_csv}")
```