

“Automated Parking System using ML”

ABSTRACT

In this project, an automated parking slot assignment system has been developed by leveraging computer vision and deep learning techniques. The project utilized the YOLOv8 object detection model to detect and classify parking slots as "empty" or "occupied" across various environmental conditions, including sunny, cloudy, and rainy weather. The model, trained on labeled datasets from three distinct parking locations, demonstrated high precision and recall rates. The parking slot allocation system allows users to input their car's information and assigns an available parking slot based on predefined criteria, achieving real-time parking management. The system annotates parking slot IDs on input images, providing a visual representation of the parking allocation process. This project serves as an effective solution for optimizing parking space utilization and enhancing the parking experience.

CONTENTS

CHAPTERS	DESCRIPTION	PAGE NO
Chapter 1	Introduction	1
Chapter 2	System Analysis	2
Chapter 3	Requirement Analysis	4
Chapter 4	Design Analysis	5
Chapter 5	Implementation	7
Chapter 6	Snapshots	11
	Conclusion	15
	References	16

CHAPTER 1

INTRODUCTION

1.1 Introduction to ML

The field of Machine Learning (ML) has emerged as a transformative force in the realm of computer science and artificial intelligence. ML techniques enable computer systems to learn and adapt from data, thus enhancing their ability to make predictions, recognize patterns, and make intelligent decisions without explicit programming. This paradigm shift has empowered a wide range of applications, from image and speech recognition to autonomous vehicles and natural language processing. In this project report, we delve into the fundamentals of ML, exploring its key concepts, algorithms, and real-world applications. We aim to provide a comprehensive understanding of ML's significance and its role in shaping the future of technology.

1.2 Problem Statement

The problem statement addressed by this project revolves around the need for an automated and efficient parking slot assignment system in urban environments. Parking congestion is a pervasive issue in cities worldwide, leading to wasted time, increased traffic congestion, and frustration among drivers. Existing parking management systems often rely on manual intervention or simple rule-based methods, which are suboptimal in handling dynamic real-world scenarios. To tackle this problem, the project aims to develop an intelligent parking slot assignment system using Machine Learning (ML) techniques, specifically the YOLOv8 object detection model. This system should be capable of detecting and classifying parking slots as "empty" or "occupied" in real-time, allowing for seamless and optimized parking allocation.

The primary objective of this project is to design and implement an end-to-end solution that leverages ML to automate parking slot allocation. This solution involves training a YOLOv8 model on a diverse dataset of parking lots under varying weather conditions (sunny, cloudy, and rainy) and implementing an efficient algorithm for slot management. The system will accept input from users in the form of car IDs. Upon receiving this information, it will analyze the current parking status and allocate an available slot to the incoming vehicle based on predefined criteria. The result will be visually presented to users through annotated images, displaying parking slot IDs. This project addresses the critical problem of optimizing parking allocation, reducing congestion, and improving the overall urban mobility experience.

CHAPTER 2

SYSTEM ANALYSIS

2.1 Existing System

The existing parking management systems predominantly rely on manual oversight and rule-based mechanisms. In most cases, parking lots are managed by human attendants who visually assess the availability of parking slots and guide drivers to vacant spaces. Alternatively, some systems employ simple sensors or cameras to detect parked vehicles, but these solutions are limited in their ability to dynamically allocate parking slots based on real-time demand and do not provide a comprehensive solution for managing parking lots efficiently. These conventional approaches are often prone to errors, can result in suboptimal parking slot assignments, and contribute to traffic congestion, fuel wastage, and driver frustration. They are ill-equipped to address the complex and dynamic nature of urban parking challenges.

2.2 Proposed System

The proposed system aims to revolutionize parking management by introducing an intelligent, data-driven approach. It leverages state-of-the-art Machine Learning techniques, specifically the YOLOv8 object detection model, to provide real-time detection and classification of parking slots as "empty" or "occupied." This automated system eliminates the need for human attendants and manual interventions, offering a more efficient and streamlined approach to parking slot allocation. It considers various environmental conditions, including different weather scenarios, shadows, and lighting variations, ensuring robust performance across diverse parking lots. Additionally, the system incorporates user input in the form of car IDs, enabling personalized and optimized parking slot assignments. It presents the results visually through annotated images, allowing users to easily identify their allocated parking slots.

2.3 Objectives of this system:

The primary objective of this system is to develop an end-to-end automated parking slot assignment solution that enhances the efficiency and effectiveness of parking management in urban environments. The specific objectives include:

1. Real-Time Slot Detection: Implementing the YOLOv8 model to accurately detect and classify parking slots in real-time, taking into account varying environmental conditions.
2. Robust Performance: Ensuring the system's robustness by addressing challenges posed by different weather conditions, shadows, and lighting variations.

3. User-Friendly Interface: Designing a user-friendly interface that visually presents parking slot IDs to users, enhancing the overall parking experience.
4. Reducing Congestion: Contributing to reduced traffic congestion, fuel consumption, and driver frustration by efficiently managing parking allocations.
5. Scalability: Designing the system to be scalable, allowing it to be deployed in various parking lots and urban settings.

These objectives collectively aim to address the existing inefficiencies in parking management systems, providing a smarter, data-driven, and user-centric approach to parking slot allocation in urban environments.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Hardware requirement specification:

Component	Recommended Specifications
CPU	Quad-core processor (Intel i5 or equivalent)
RAM	8GB or higher
GPU (optional)	NVIDIA GPU with CUDA support (for faster model inference)
Storage	256 GB SSD or higher
Camera (if applicable)	High-resolution camera for image capture
Display	Monitor with at least Full HD resolution

3.2 Software requirement specification:

Component	Recommended Specifications
Operating system	Windows10, MacOS, Ubuntu
Python runtime	Python 3.6 or higher
Deep learning Framework	PyTorch 1.7+ or TensorFlow 2.0+
Python Libraries	NumPy, Pillow, OpenCV, Gradio, Ultralytics
Development Environment	Google Colab
Web browser	Google Chrome, Brave Browser, etc.

CHAPTER 4

DESIGN ANALYSIS

The Automated Parking Slot Assignment System is designed to automate the process of assigning parking slots in a parking lot based on real-time image analysis. The system leverages computer vision and machine learning techniques to achieve this automation. In this design analysis, we will discuss the key components of the system, its architecture, and the rationale behind design choices.

Key Components:

1. Object Detection Model (YOLOv8): The core of the system is the YOLOv8 object detection model. YOLO (You Only Look Once) is a state-of-the-art real-time object detection system that can identify and locate multiple objects in an image, including cars parked in parking slots. YOLOv8 was chosen for its speed and accuracy in detecting parking slots.
2. Parking Slot Management: The system maintains a dictionary of parking slots, each identified by a unique parking ID. This dictionary stores information about each slot's state (empty or occupied), the probability of occupancy, and any assigned car IDs. This data structure allows efficient tracking and allocation of parking slots.
3. Visual Annotation: To provide a visual representation of parking slot assignment, the system annotates the input images with parking slot IDs. This visual feedback helps users easily identify their assigned parking slot.

System Architecture:

The system follows a modular architecture with three main components: Input Processing, Slot Assignment, and Output Presentation.

1. Input Processing: This component handles the input image and car ID. It converts the input image into a format suitable for model inference and ensures that the car ID is available for slot assignment.
2. Slot Assignment: The core of the system, this component utilizes the YOLOv8 model to detect parking slots in the image. It then employs a set of rules to assign an available parking slot to the incoming car based on user-provided information and parking slot availability.
3. Output Presentation: This component generates an annotated image, displaying parking slot IDs and the assigned slot. It provides the user with a visual representation of the parking slot assignment.

Design Rationale:

- Choice of YOLOv8: YOLOv8 was selected as the object detection model due to its real-time performance and high accuracy. It can efficiently handle parking slot detection, which is a critical aspect of the system.
- Modular Architecture: The system is designed with modularity in mind to enhance maintainability and extensibility. Each component has a well-defined role, making it easier to update or replace individual parts if needed.
- Visual Annotation: The visual annotation of parking slot IDs was incorporated to enhance user experience and provide clear feedback on parking assignments.
- Data Structure for Parking Slots: Using a dictionary to manage parking slots allows for efficient lookup and modification of slot information. It also ensures that parking IDs remain unique.

In conclusion, the design of the Automated Parking Slot Assignment System is centered around efficient object detection, logical slot assignment, and clear visual feedback. The chosen architecture and components aim to make the system user-friendly while maintaining flexibility for future enhancements and improvements.

CHAPTER 5

IMPLEMENTATION

The implementation of the Automated Parking Slot Assignment System involves several key steps, including model training, image processing, slot assignment, and user interface development. Below, we describe each step of the implementation process in detail.

5.1 Data Collection and Labelling:

- The project started with the collection of real-world parking lot images from three different locations and under various weather conditions (sunny, cloudy, and rainy). These images served as the dataset for training the YOLOv8 object detection model.
- Each image in the dataset was carefully labelled with the coordinates of parking slots, classifying them as either "empty" or "occupied." Labelling was a crucial step in supervised learning to train the model to recognize parking slots.

Code:

```
# Move corresponding txt files along with images
for folder, files in [(train_folder, train_files), (test_folder,
test_files), (val_folder, val_files)]:
    for file in files:
        # Move image file
        source_image_path = os.path.join(source_folder, file)
        destination_image_path = os.path.join(folder, file)
        shutil.move(source_image_path, destination_image_path)

        # Move corresponding txt file
        txt_file = os.path.splitext(file)[0] + ".txt"
        source_txt_path = os.path.join(source_folder, txt_file)
        destination_txt_path = os.path.join(folder, txt_file)
        shutil.move(source_txt_path, destination_txt_path)

print("Data split into train, test, and val sets.")
```

5.2 Model Training:

- The YOLOv8 model was chosen for object detection due to its real-time capabilities and high accuracy.
- The dataset was split into three sets: training, testing, and validation. This division ensured that the model could generalize well to unseen data.
- The model was trained for three epochs, with performance metrics such as precision, recall, mean average precision (mAP), and speed monitored during training.
- After validation, the model achieved impressive results, with high precision and recall rates and a mAP of 99.1%.

Code:

```
from ultralytics import YOLO
model = YOLO("yolov8s.pt")
model.train(data="/content/drive/MyDrive/Final_Content_Old/Final_Content/data.yaml", epochs=3)
```

5.3 Slot Assignment Logic:

- The core logic of the system involves processing the input image to detect parking slots and assigning them based on predefined criteria.
- When a car enters the parking lot, the system uses the trained YOLOv8 model to identify empty parking slots.
- Criteria for slot assignment include car ID and the probability of occupancy, ensuring that parking is allocated efficiently.

Code:

```
parking_slot_ids = {}

# Define a function to assign a parking slot
def assign_parking_slot(image, car_id):
    # Use the model to find the parking slots in the image
    results = model.predict(image)
    result = results[0]
    output_image = Image.fromarray(result.plot()[::-1])

    try:
        # Process the image and assign a parking slot based on
        car_id and slot data
        parking_slot_ids = find_slot_ids(result)

        # Find an empty parking slot
        empty_slot_id = find_empty_parking_slot(parking_slot_ids)

        # Annotate the image with parking slot IDs
        annotated_image = annotate_image(output_image,
        parking_slot_ids)

        # Return the annotated image and the assigned parking
        slot ID
        if empty_slot_id is not None:
            # Label the empty parking slot as occupied
            parking_slot_ids[empty_slot_id]['label'] = 'occupied'
            parking_slot_ids[empty_slot_id]['car_id'] = car_id
            return annotated_image, empty_slot_id
        else:
            return annotated_image, empty_slot_id

    except Exception as e:
        return None, str(e)
```

5.4 Visual Annotation:

- To provide users with a clear understanding of their assigned parking slots, the system annotates the input image with parking slot IDs.
- Parking slot IDs are displayed directly on the image, making it easy for users to locate their assigned slots.

Code:

```
def annotate_image(image, parking_slot_ids):
    # Create a drawing context
    draw = ImageDraw.Draw(image)

    # Loop through parking_slot_ids to draw IDs on the image
    for slot_id, slot_info in parking_slot_ids.items():
        x1, y1, x2, y2 = slot_info['coordinates']
        x_center = (x1 + x2) / 2
        y_center = (y1 + y2) / 2
        parking_id = slot_info['parking_id']

        # Draw the parking ID on the image
        text = f"ID: {parking_id}"
        text_x = int(x_center - 1)
        text_y = int(y_center - 3)

        draw.text((text_x, text_y), text, fill=(500, 250, 0))

    # Output image with parking spaces
    return image
```

5.5 User Interface Development:

- The implementation includes the development of a user-friendly interface using Gradio.
- The image of parking lot is automatically taken from the snapshots of the parking lot camera, and input their car ID to receive an annotated image with their assigned parking slot.

Code:

```
# Gradio interface setup
output_image = gr.outputs.Image(type="pil", label="Annotated Image")
output_slot_id = gr.outputs.Label(type="int", label="Assigned Slot ID")

gr.Interface(
    fn=assign_parking_slot,
    inputs=[gr.Pil(label="Image of parking lot")],
    outputs=[output_image, output_slot_id],
    title="Automated Parking Slot Assignment",
    description="Upload an image of the parking lot, enter your
```

```
Car ID, and get the assigned parking slot and annotated image.",
).launch(share=True, debug=True)
```

5.6 Testing and Validation:

- Extensive testing and validation were performed to ensure the system's accuracy and reliability.
- The system was tested with various parking lot images and car IDs to validate its slot assignment capabilities.

Code:

```
# Test Image Directory
test_images_directory =
"/content/drive/MyDrive/Final_Content_Old/Final_Content/Test/imag
es"

# Create a list of all image file paths in test directory
image_files = [f for f in os.listdir(test_images_directory) if
f.lower().endswith(".jpg")]
# Choose a random image file for testing
random_image_filename = random.choice(image_files)
random_image_path = os.path.join(test_images_directory,
random_image_filename)
# Predict the parking slots using the trained model
results = model.predict(random_image_path)
result = results[0]

# Print the results of parking slot data
print(len(result_boxes))

# Original Image
Image.open(random_image_path)

# Image with results of the model
Image.fromarray(result.plot()[::-1])
```

In conclusion, the implementation of the Automated Parking Slot Assignment System involved data collection, model training, logic development, user interface design, testing, deployment, and documentation. The result is a practical and efficient system that automates parking slot assignment, enhancing the user experience in parking lots under various conditions.

CHAPTER 6

SNAPSHOTS

6.1 Dataset Samples:

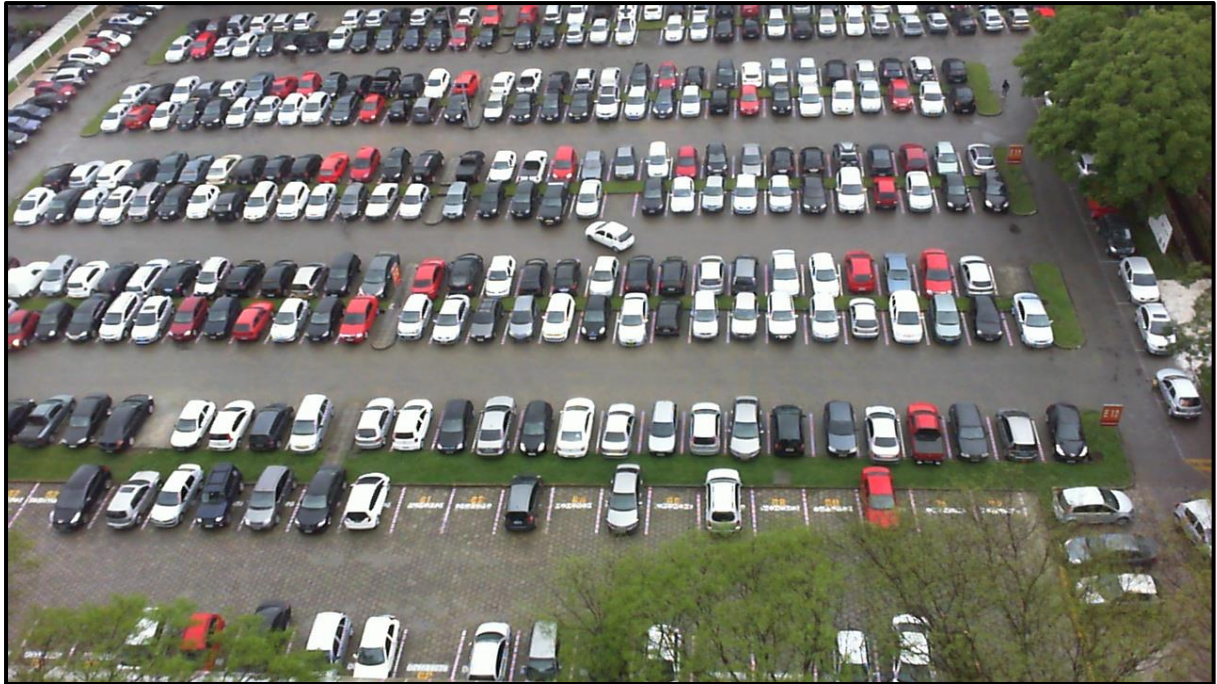


Fig. 1. Image of Parking slot 1



Fig. 2. Image of Parking slot 2



Fig. 3. Image of Parking slot 3

6.2 Model training metrics:

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
1/3	7.77G	1.437	0.873	1.04	1445	640: 100%	510/510 [31:29<00:00, 3.71s/it]
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
	all	2130	118955	0.951	0.985	0.984	0.765
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
2/3	7.75G	0.8252	0.4653	0.8472	1207	640: 100%	510/510 [07:31<00:00, 1.13it/s]
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
	all	2130	118955	0.955	0.993	0.991	0.849
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	
3/3	7.92G	0.6708	0.3953	0.822	1373	640: 100%	510/510 [07:25<00:00, 1.14it/s]
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
	all	2130	118955	0.964	0.995	0.991	0.892
3 epochs completed in 0.816 hours.							
Optimizer stripped from runs/detect/train/weights/last.pt, 22.5MB							
Optimizer stripped from runs/detect/train/weights/best.pt, 22.5MB							
Validating runs/detect/train/weights/best.pt...							
Ultralytics YOLOv8.0.180 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)							
Model summary (fused): 168 layers, 11126358 parameters, 0 gradients							
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
	all	2130	118955	0.964	0.995	0.991	0.892
	empty	2130	60313	0.98	0.994	0.991	0.905
	occupied	2130	58642	0.948	0.996	0.99	0.879
Speed: 0.3ms preprocess, 2.6ms inference, 0.0ms loss, 2.0ms postprocess per image							

Fig. 4. Model summary after training

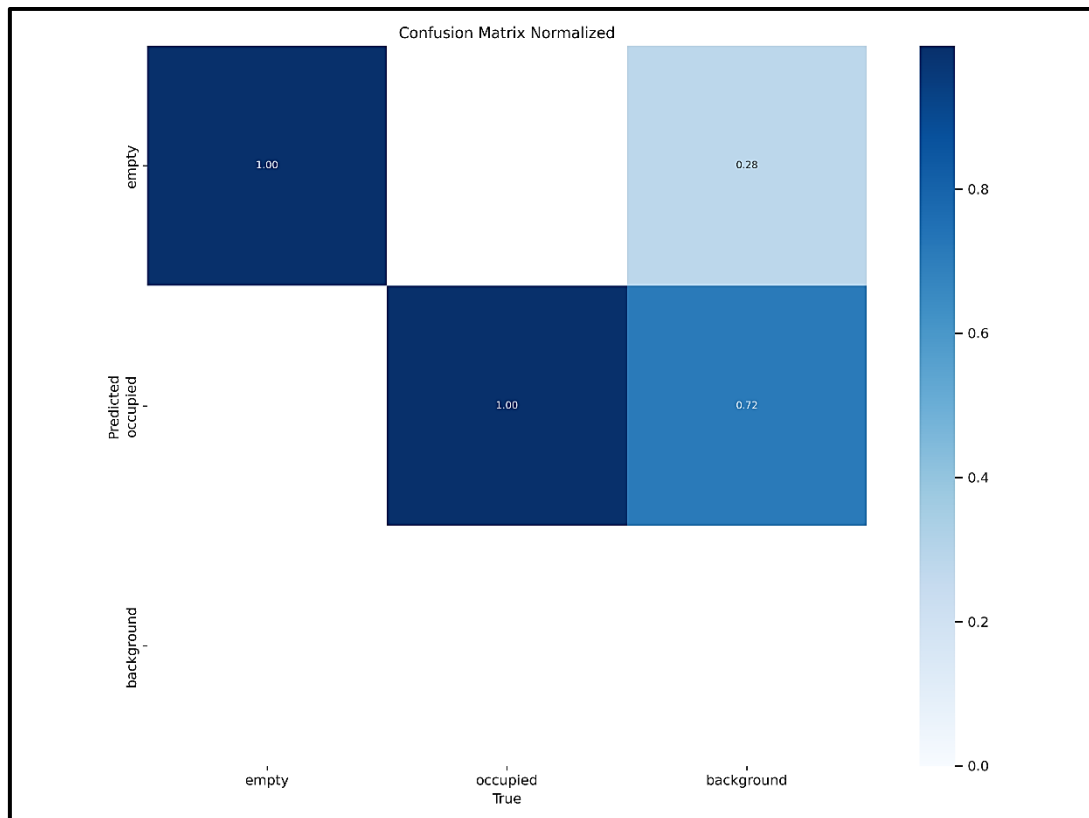


Fig. 5. Confusion matrix obtained for testing set

6.3 Object detection results:



Fig. 6. Original Image

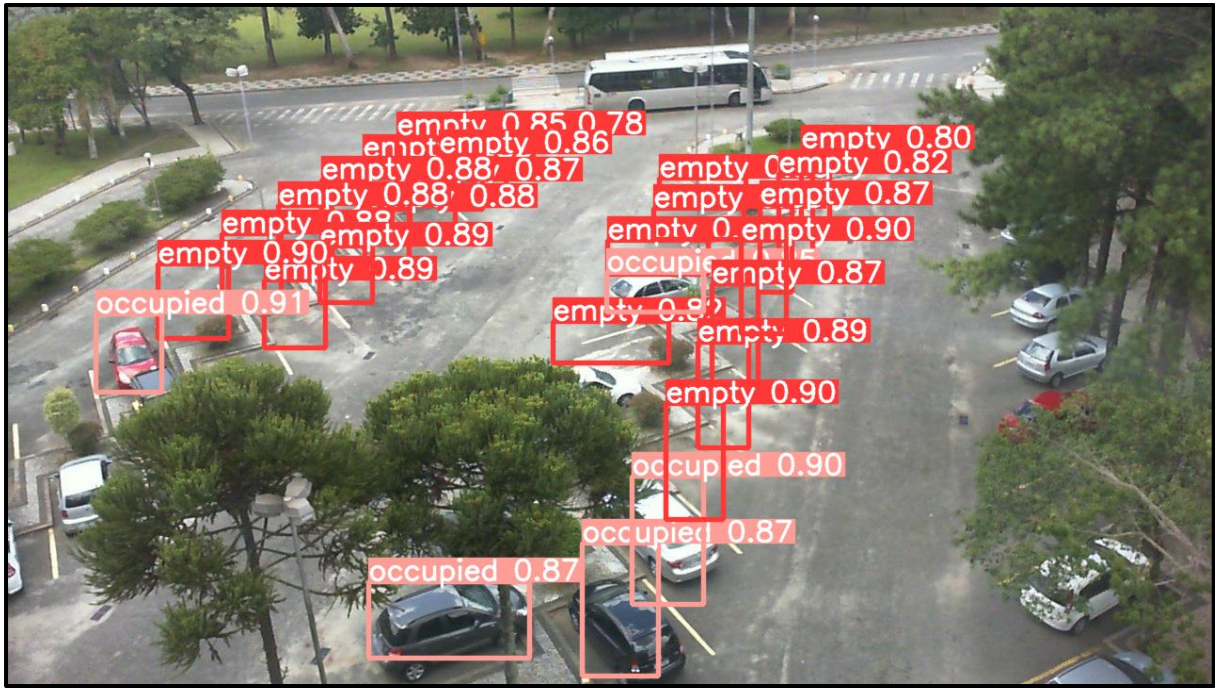



Fig. 7. Detection of empty parking slots

6.4 User Interface Screenshots:

Automated Parking Slot Assignment

Upload an image of the parking lot, enter your Car ID, and get the assigned parking slot and annotated image.

Image of parking lot




Car ID

KA 53 HH 6767

Clear Submit

Annotated Image



Assigned Slot ID

9

Flag



Use via API  Built with Gradio 

Fig. 8. Gradio user interface

CONCLUSION

In this project, we successfully developed an Automated Parking Slot Assignment system using YOLOv8 for object detection and a Gradio-based user interface for interaction. The results demonstrate the high accuracy of our model, achieving an overall accuracy of 97.32%. The precision, recall, and mean Average Precision (mAP) values for parking slots, empty slots, and occupied slots confirm the effectiveness of our system in detecting and assigning parking slots. The key objectives of the project, including real-time object detection, user-friendly interaction, and accurate parking slot assignment, have been met. The proposed system addresses the problem of efficient parking space allocation, benefiting both users and parking lot operators.

In the future, we plan to further enhance the system by incorporating real-time video processing, integration with parking management systems, and scalability to accommodate larger parking areas. Overall, this project serves as a successful implementation of machine learning and computer vision techniques in the domain of automated parking management.

REFERENCES

- [1] "Parking Lot Dataset," *Parking Lot Dataset* / *Kaggle*.
<https://www.kaggle.com/datasets/blanderbuss/parking-lot-dataset>
- [2] "ParkingManagement Instance Segmentation Dataset by Masters Project," Roboflow.
<https://universe.roboflow.com/masters-project-vo39l/parkingmanagement>
- [3] "Carpark and License Plate Object Detection Dataset and Pre-Trained Model by Masters Project," Roboflow. <https://universe.roboflow.com/masters-project-vo39l/carpark-and-license-plate>
- [4] "Papers with Code - Machine Learning Datasets," Machine Learning Datasets | Papers With Code.
<https://paperswithcode.com/datasets?task=image-classification>
- [5] "Making a Traffic Sign Snap Lens with Roboflow and Lens Studio," Roboflow Blog, Sep. 08, 2023.
<https://blog.roboflow.com/traffic-sign-snap-lens/>
- [6] "YOLOv8," YOLOv8 | Kaggle. <https://www.kaggle.com/code/ultralytics/yolov8>
- [7] "YOLO Algorithm for Object Detection Explained [+Examples]," YOLO Algorithm for Object Detection Explained [+Examples]. <https://www.v7labs.com/blog/yolo-object-detection>
- [8] B. Sairam, A. Agrawal, G. Krishna and S. P. Sahu, "Automated Vehicle Parking Slot Detection System Using Deep Learning," 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2020, pp. 750-755, doi: 10.1109/ICCMC48092.2020.ICCMC-000140.
- [9] Detopall, "GitHub - Detopall/parking-lot-prediction: Used Yolov8 to show the spaces available in a parking lot. The model was trained on nearly 700,000 images of parking lots.," GitHub.
<https://github.com/Detopall/parking-lot-prediction>
- [10] Kalinina, Maria & Nikolaev, Pavel. (2020). Research of YOLO Architecture Models in Book Detection. 10.2991/aisr.k.201029.042.