# Vehicle detection and identification of free slots in a garage

by

Soumitra Das
20216006
Shoabur Rahman Chishty
22101312

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

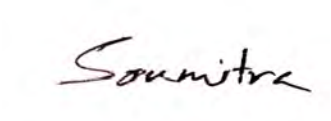
Department of Computer Science and Engineering
Brac University
October, 2024

# Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

**Student's Full Name & Signature:**



Soumitra Das

20216006

Shoabur Rahman Chishty

22101312

# Approval

The thesis/project titled "Vehicle detection and identification of free slots in a garage" submitted by

1. Soumitra Das (20216006)
2. Shoabur Rahman Chishty (22101312)

Of Summer, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 22, 2024.

**Examining Committee:**

Supervisor:
(Member)

Mr. Dibyo Fabian Dofadar

Lecturer
Department of Computer Science and Engineering
BRAC University

Co-supervisor:
(Member)

Mr. Rafeed Rahman

Lecturer
Department of Computer Science and Engineering
BRAC University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi

Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

# Abstract

The illegal or improper parking of vehicles interferes with the flow of traffic, and represents a potential hazard to both pedestrians and drivers, worldwide. Parking in unsuitable locations can also cause damage to the vehicle, so garages are the safer choice of parking for everyone. But the hard reality of the situation is that garage owners are not always skilled at efficiently handling the available parking slots for customers, due to the lack of experienced staff and drivers' unaware of vacant spaces. Some developed countries use Internet of Things (IoT) based solutions and CCTV installations in managing parking but are very costly. Understanding these difficulties, our research endeavors to develop a cost efficient method for the detection and identification of free parking slots in garages by means of machine learning techniques. Currently, machine learning models can educate vehicle detection successfully, but they are insufficient in recognizing free parking spaces. In this study we used various algorithms such as Faster R-CNN, YOLO, and MobileNet SSD to find out which model is most suitable when we wanted to detect both vehicles and free slots accurately. After selecting the model, we expanded upon the detection process by adding on more steps to the model. Using only existing CCTV footage from the garage, our proposed algorithm does not require additional IoT devices for training and deployment, thereby significantly reducing costs for owners and providing convenience for users. On the contributing side of this research, it addresses the optimization of parking resources for various scenarios, and helps to improve both parking related services and the overall user experience. Additionally, in free space detection reliability increases to enable the establishment of parking services, which is identified as the main problem for garage users: the free space availability. Thus, a simplified management system may start to encourage private garage owners to open their facilities to the public. Finally, this research provides a useful solution to parking problems on a global scale.



**Keywords:** Faster R-CNN, YOLO, MobileNet SSD, IOT, CCTV Camera

# Table of Contents

# Chapter 1

# Introduction

## 1.1 Thoughts behind the Model

Parking cars or vehicles is a challenge both for car owners as well as for drivers, especially in big cities and crowded large urban areas. From January, 2024 to September, 2024, the number of registered motor vehicles is 222,918 and the grand total is 6,175,838, says the Bangladesh Road Transport Authority (BRTA) report [27]. These figures show the increase in the rate of vehicle usage in Bangladesh with rising numbers in vehicles. The growing population and increasing number of vehicles mean that the quantitative availability of organized parking spaces is becoming a problem. In urban areas, where people are always locked within tight schedules, appropriate parking facilities are necessary to keep vehicle owners safe and convenient. According to CEIC Data in 2022, Dhaka, Bangladesh, had approximately 175,000 registered motor vehicles [16]. However, about only 80 percent of motor vehicles are registered [1], indicating that there are many more vehicles than that. They haven't scuttled away to someplace nice and safe like the hubs in other cities; instead, there's free parking available all over the city sans a reliable way to find it. Due to this, most drivers have no option but to park their vehicle on roadsides, which is not very friendly for either the drivers or pedestrians or traffic congestion. This is the more worrying issue when it comes to countries that are not yet developed or developing and roads and transportation systems are unevenly and often misguided. Parkings are scattered and the drivers do not have proper parking spaces as a result of which they park their cars with no regard to law, there is chaos created and traffic jams get worse. Additionally, in cities like Dhaka (Bangladesh's capital), for instance, parking is risky because of political unrest. This will make parking vehicles in a garage safer, and more beneficial to their owners as vehicles are valuable assets for their owners. Finding free parking spaces inside a garage every day is time consuming and inefficient — especially when you live or spend time in a busy city where people tend to be in a rush. Most countries do not have an organized platform where the large number of vehicles can find enough temporary parking spaces and to be able to track available spaces in real time. Although some use manual or automated methods to monitor open spaces, their costs may outweigh ours. Generally, traditional garage owners must hire skilled personnel to monitor the number of vehicles entering and exiting the facility in order to detect free spaces. Tracking, collecting and often updating this data can be expensive, requiring use of multiple staff members. For example, in their garages, developed countries have developed solutions like Internet

of Things (IoT) devices. This too is an expensive option, because the owner needs to install hardware for every parking space. In addition, there are consequent costs for the installation of CCTV cameras to secure places. These measures therefore contribute a small return on investment for public or private garage owners. As a result, neither owners nor government officials are stimulated to create additional public or private garages as personal vehicles grow larger. Using CCTV being fundamental in garages for their security purposes, it seems natural to tap into existing systems. In the era of scientific development, it is no longer possible in terms of implementing additional IoT devices or more personnel to implement the same. Therefore, as means of parking space monitoring, it's quite innovative, reliable and cost effective to be using existing CCTV cameras. The consideration of this problem requires the development of a low cost solution employing machine learning algorithms to automate the data collection from CCTV cameras. This would decrease the amount of need for large funding and excessive man hours. The algorithm is necessary to work well even for any kind of data gathered from garage CCTV footage, giving an immediate count of the number of vehicles at the moment and how many parking spaces are available. It will allow people to know where they can park their cars safely. With minor modifications, the algorithm can detect cars and empty slots in real time, and efficiently detect them for free spaces in real time. They make it possible to deploy IoT devices without the need for excess manpower and a huge budget in the creator's hands. Not only will this solution be convenient to parking for vehicle owners, but it will also substantially reduce the cost and labor associated with garage ownership. In the end, this derived approach seeks to improve parking resource utilization and the parking experience in urban areas.

## 1.2 Problem Statement

Parking management detection technologies based on machine learning could be divided into two major tasks. The very first task is detecting accurately the vehicles that are present in the garage at any moment in time. The second and more difficult problem is to predict the number of free parking slots available. The second task is complicated because it is hard to determine free spaces without consideration to other existing vehicles inside the garage. To overcome this challenge, it is necessary to have a system that not only can detect vehicles, but also empty slots in a garage environment. Generally, we calculate the accuracy of free space detection from the previous arrangement of cars. The system identifies and predicts the boundary boxes of potential free spaces by using information about previous vehicle placements. The problem is made complicated by the fact that these bounding boxes are bare (in other words, vacant) and without particular distinguishing features that the model will train itself on. Our detection model relies on the relative positions of the parked vehicles to locate free slots. However, because of varying distances between vehicles and differing orientations we can have the same number of vehicles take up different amounts of space, which can lead to lower accuracy of this approach. However, these variations result in it being hard for the model to always predict the free spaces solely on the basis of vehicle positions. Consequently, improvements in machine learning algorithms used in free space detection are required. Incorporating enhanced methods for free space detection are the main goal of this research to solve the mentioned challenges. Along with the vehicle detection accuracy, this study also

tries to push forward the free space detection technology to new levels of precisions and could help better parking management solutions.

## 1.3 Aims and Objectives

The objective of this research is to increase the ease of use, operability, and reliability for garage proprietors and vehicle owners of garage parking systems. In this study we seek an algorithm that uses currently available CCTV footage and provides as accurately as possible free parking space detection and identification, with negligible error. This research aims to help address parking challenges around the world by creating a predictive model that virtually mirrors human perception of empty slots, in order to easily and accurately pick a place to park. The research also attempts to reduce the dependence on huge funding and manpower required to establish a viable platform for car parking information knowledge base. We aim to offer a cost effective and efficient solution for parking resource utilization with advanced machine learning techniques capable of improving user experience. Not only will this solution help to solve parking problems, but this approach will facilitate the spread of technology to enhance the use of technologically sophisticated systems employed in managing parking, in turn improving urban infrastructure and transportation efficiency.

## 1.4 Contribution

The reason IoT based solutions are considered more reliable than machine learning based solutions because it gives almost 100% precision in detecting vehicles and empty slots. But machine learning based solutions provides more flexibility for the garage owners. Our research will help to improve the reliability of already existing machine learning model implemented in a garage by trading off execution time. It also will help to develop a customized model for a particular garage easily. We used YOLOv8 model to demonstrate how it can be done, testing it with both a small and large sized datasets.

# Chapter 2

# Literature Review

In this chapter, we introduced an extensive literature review of various research paper to understand the current state of the art of vehicle detection and parking slot identification. Specifically, it provides the foundational knowledge to choose appropriate machine learning models and techniques most suited to address challenges presented by existing solutions. Note that their implemented models were not used directly for our research purpose.

From 3D Object Detection and Tracking Methods using Deep Learning for Computer Vision Applications [11], we get a basic overview of current technologies and algorithms. This article addressed different problems and limitations regarding performance metrics of 3D object detection technologies and methods. It is not a research paper targeting a specific problem, but a collection of improvement history using LiDAR, RGB, point cloud or different types of data from different researchers. Then it mentioned different available datasets for 3D object detection like CamVid, Objectron, KITTI, etc. Then it introduced different object detection methods for different kinds of datasets, for example: 3D-GCK for RGB images, DPC-MN for point clouds, etc. Then it described GNN, Edged based, QCNN methods of detection of tracking. Then it introduced different evaluation metrics like sAMOTA, AMOTA, AMOTP, IDS, FRAG and FPS. Then it described different applications of object detection in 3D space, which are augmented reality, autonomous driving, robotic systems which arevision guided, etc. Then it introduced possible applications of 3D object detection and who are using them. Then it addressed some challenges like data collection, availability, and computing speed problems. So, this article gave an overview of 3D object detection and tracking methods and challenges.

Next, we see research, Computer Vision Based Object Detection and Recognition System for Image Searching[14] which works on a binary classification model. This research paper proposes a system (an image search engine) that detects and recognizes objects from images using machine learning and claims that the system achieves the highest accuracy rate of 99.98% and the lowest accuracy rate of 89.28%, which are better than the previous state-of-art. Their system uses TensorFlow which uses CNN to train a large dataset of 15 million images with more than 21000 classes from ImageNet. Their dataset is collected from inception-v3 and it is classified. The paper also mentions previous works on object detection and recognition us-

ing deep learning methods, such as CNN, OverFeat, R-CNN, DNN, and RNN. The paper compares the accuracy rates of different methods on various datasets, such as Pascal VOC, MIT-67, and MNIST. The paper also discusses the challenges and limitations of the existing methods, such as overfitting, segmentation, and captioning. The proposed system first classifies the dataset. After completing training, it creates a protocol buffer file. The system decodes, parses, and matches it according to the class after providing the input images. At last, the best guess is returned as the result. The paper also explains the equations and metrics for object localization and detection using the ILSVRC-2017 challenge.

The binary classification is done based on vehicles in the research paper -A hybrid deep learning CNN-ELM approach for parking space detection in Smart Cities[17], which also tries to solve problems about vacant space detection due to camera positioning, weather conditions, obstacles, etc. A hybrid deep learning model, which combines CNN and ELM to classify the parking spaces where CNN is used to extract feature from the images, and ELM is used to accurately classify the images is proposed in the research paper. This study identifies the problem of automatically detecting vacant or occupied parking spaces in smart cities using images captured by cameras which is a challenging task due to the varying weather conditions, camera views, and obstructions that affect the image quality and visibility of the parking lots. A dataset collected from PKLot is used to evaluate the proposed model, which contains around 700,000 parking lot images with different camera views and weather conditions. The paper proposes a CNN-ELM model that consists of two phases: feature extraction where they use a CNN model based on LeNet-5 architecture to extract local patterns from the input parking lot images and classification where they replace the fully connected layer of CNN with a ELM model and perform fast classify the parking space as vacant or occupied fast and accurately. Furthermore, some of the recent works in this field is discussed here. The proposed CNN-ELM approach uses various performance metrics such as accuracy, AUC score, sensitivity and specificity to assess the effectiveness of their approach and compare it with modlels of the similar type. The experimental results show that proposed model outperforms other methods in terms of accuracy, sensitivity, and specificity in eight out of nine cases, and achieves comparable results in the remaining case.

Next, we see a research, Vehicle Detection and Recognition[9], which goes a bit farther. The target of this paper is to introduce the processing of automatic vehicle detection and recognition using CCTV footage images and then implement License Plate Recognition (LPR). The static image is processed with MobileNet SSD model and outputs location of vehicles and their types. The LPR is done by scraping the plate text image and using OCR to convert it to text. Corresponding research review is described, including research by Tang. Y. Zhang and Team (2015, on vehicle detection, 97% accuracy), Priyanka Prabhakar and Team (2014, on LPR, 94.34% accuracy) and Saran K.B and Team (2015, classification accuracy 75.1%). The method is run in two steps, one is vehicle detection, which outputs a location, followed by vehicle recognition, which outputs vehicle type. They sequentially used Haar-like features, AdaBoost algorithm for detection, then Gabor's wavelets transformation, Histogramic Sequencing, then Principal component Analysis for vehicle recognition. For LPR, the sequences are detecting vehicles, searching

for license plate frames, sharpening images, shmooten rectangular number plates, localization, character segmentation and finally character recognition. The proposed hybrid model has accuracy of 95.6% with a false positive rate of 0.0389.

Next, Vehicle detection and tracking[12] research paper increases both the bounding box accuracy and output accuracy. The paper proposes some more efficient features like feature extraction, model training and assessment, sliding-window technique and pipeline on a video stream for vehicle detection and tracking. The proposed system uses machine learning techniques to detect and track vehicles efficiently in images and videos. It uses three feature extraction methods which are spatial binning, color histogram, and HOG. It has trained a LinearSVC, a classifier from support vector machines (SVMs), to predict vehicles in a dataset of 8792 vehicle images and 8968 non-vehicle images. Moreover, it uses stratified 10-fold cross validation and the area under the ROC curve (AUROC) to assess the model performance. To search vehicles in the images, sliding window technique is proposed in this paper. It also uses HOG sub-sampling, thresholding and heatmap to combine overlapping detections and remove false positives. The head map and thresholding methods are used to combine overlapping detections and remove false positives. Thus the result gives more accurate bounding boxes for the vehicles. After processing each frame of the video images with hog sub-sampling and adding a bounding box for vehicles to detect, the system creates a pipeline video. Additionally, the system has achieved 99.37% test accuracy of the LinearSVC classifier . Furthermore, the area under the ROC curve (AUROC) for the LinearSVC classifier was $0.99 \pm 0.01$. AUROC LinearSVC classifier result indicates the higher true positive rate and the lower false positive rate. Again, the hog sub-sampling technique not only improved the efficiency but also speed of the sliding window approach, reducing the calculation time for finding HOG features. Similarly, the pipeline video showed the successful detection and tracking of vehicles in a video stream.

Next, Transfer Learning with Image Data Augmentation for Parking Occupancy Detection[18] paper claims to solve parking occupancy detection with high accuracy with low computational parameters, which works for different weather conditions and parking lot layouts. The paper proposes a new method based on transfer learning and image data augmentation to classify parking spaces from images captured by cameras identifying the problem of parking in the city as well as the need for efficient parking management solutions. The goal was to detect vehicle parking spaces. The paper surveys previous works on image classification for parking occupancy detection. It not only focuses on the use of convolutional neural networks (CNNs) and transfer learning but also compares with some different architectures of CNN, such as CarNet, AlexNet, VGGNet, ResNet, and Inception, and their performance on the PKLot and CNRPark-EXT datasets mentioning the advantages and challenges of using transfer learning for this task. The paper proposes a model architecture using transfer learning based on the Inception V3 which uses data augmentation with some modifications and their dataset was PKLot. The final layers of the pre-trained model replaces with custom prediction layers, including a GlobalAveragePooling2D layer, two dropout layers, two dense layers having batch normalization, and a final dense layer with sigmoid function activation. The image data augmentation technique was used to increase the size, improve the accuracy of the CNN model, and

diversity of the dataset. The experiment introduces the three subsets (PUCPR, UFPR04, and UFPR05) of the PKLot dataset divided into training, validation, and testing sets. After comparing with other models from the literature the result was the proposed model achieves high accuracy with low computational parameters and is robust to different weather conditions and parking lot layouts. Their result shows high accuracy because maybe they have fewer parameters than other models.

Next, paper Vehicle Counting System in Urban Areas: A Practical Case[13] shows a way to detect and count vehicles at a low cost and low resource consumption. A new functional way of counting vehicles is proposed in this paper. The paper aims to solve the problem of urban traffic congestion by providing a reliable and low-cost method of counting vehicles on roads using video analysis. They reviewed previous works where Ferreira's work was on identifying and counting vehicles from videoed which were recorded previously. Tang et al. discuss significant limitations arising from alterations in both lighting conditions and variations in vehicle speeds. Kadikis et al. employ a straightforward bar in pre-recorded videos to tally the number of vehicles passing it. Allamehzadeh et al. achieved an impressive 94% accuracy in vehicle detection. Shirazi et al. accomplished vehicle counting accuracy ranging from 89% to 94%. The paper claims that the prototype achieved an average accuracy of 93.26%. The paper also demonstrates the feasibility of running the system on a Raspberry Pi device. Prototype was developed using agile methodology through seven iterations. In the first iteration, they built a simple system that takes raw videos as input and gives the counts of vehicles as output. The prototype uses background subtraction with the help of some image processing techniques, such as erosion, dilation, and thresholding, to improve the detection in a video. Here, the Mixture of Gaussian (MOG) algorithm shows a lower 11.86% relative error than others. But it was giving not only false positives but also false negatives, as well as multiple vehicles detections. In the second iteration, they applied a strategy by drawing on the playback screen a straight line and on the detected vehicles they are plotting a point. When any vehicles crosses the straight line, the system counts them. It reduces false positives because of the not moving vehicle. In the third iteration, the algorithmic behavior was revised using a wider line on the screen and then they used a special variable to mark in each object which was detected previously to ensure no multiple counting happens while crossing the straight line. Here, the efficiency has risen to 92%. In the fourth iteration, some improvements happened like marking car detection area with two straight lines, MOG was replaced by MOG2, etc. Detected vehicles will be counted after crossing the first straight line, but will not be tracked anymore after crossing the second one. Some learnings gathered like holes and camera movement affect the accuracy. Moreover, The better the resolution, the better the accuracy. In the fifth iteration, a region of interest (ROI) was proposed to apply in real situations. Apart from some common errors, after testing on 8 different videos, 93.26% accuracy was achieved. In the sixth iteration, a graphical user interface (GUI) is created and for future analysis, a storing functionality is added. In the seventh iteration, some libraries changed and were implemented in a Raspberry Pi 4.

The paper Deep learning-based parking occupancy detection framework using ResNet and VGG-16[36] tries to detect vehicles in outdoor environments in different weather

conditions. This research proposes a solution of deep learning-based parking occupancy detection framework. They used two pre-trained convolutional neural networks which are ResNet and VGG-16. The primary objective is to establish a framework that offers an economical and dependable solution for managing parking systems in outdoor environments. The study recognizes that sensor-based techniques are not feasible in the real world. Studying different papers the research took a different approach. At the beginning, the proposed model is using pre-trained CNNs to extract features from parking lot images and classifying them as either vacant or occupied. Depending on the results, it compares their performance with ResNet50 and VGG16 to evaluate on the dataset. The solution is based on two deep learning models which are ResNet50 with an SVM classifier and VGG16 with OpenCV. First model uses a pre-trained ResNet50 for extracting features from the images of given parking spaces. Then it starts to train a SVM classifier so that it can classify the vacancy status of each parking space. After processing, the first model achieved 98.9% accuracy on the PKLot dataset. The second model uses a pre-trained VGG16 which is also for extracting features from the images of parking spaces. Then it applies OpenCV functionalities such as thresholding, contour detection, and morphological operations to segment and classify the parking spaces. The second model also achieved 3.4% accuracy on the same dataset. Moreover, the paper reviews various previous works and techniques related to parking occupancy detection. In those works, they are using image-based or camera-based systems, such as CNN, SVM, MRF, YOLO, Mask R-CNN, etc. The PKLot dataset is used to conduct the research containing 12,417 images of parking areas and 695,899 images of parking areas not only segmented but also labeled under different weather as well as illumination conditions.

But the paper, Revising Deep Learning Methods In Parking Lot Occupancy Detection[19], tries to generalize the input even farther. In this paper, authors mentioned that most of the parking space occupancy detection problem depends on specific observation angles, weather conditions, illumination intensity, missing of perspective distortions and overlaps. So a generalized solution is proposed here, which will detect vehicles regardless of those conditions fulfilled or not based on EffiientNet-B0 architecture, and applied it to datasets on 5 courtyard parking spaces, and removed biasness by geometric transformations like flip, rotation, random crop, translation, and noise insertion. Then it described several classic CNN methods for patch based classification like AlexNet, CarNet, EfficientNet, etc and intersection based classification like Faster R-CNN and RatinaNet. Then data representation methods for path and intersection based classifications and data source characteristics are mentioned. It introduced a custom EfficentNet-P method based on the original EfficientNet-B0 method by modifying MBConv1 and MBConv6 on inputs. Then it compared the results obtained from modified methods from already existing methods for both patch based and intersection based methods. According to the result obtained, EfficientNet-P showed the most accurate values for 4 out of 5 datasets.

In previous articles, we read research about vehicle detection and recognition. But the next article, Predicting Parking Occupancy with Deep Learning on Noisy Empirical Data[20], tries to predict vehicle occupancy in a particular area where all the data about all the garages are known for a certain time. This research is applied on

data of 11 parking lots of a small city of Germany, Ingolstadt, which was collected over a year. The research tried to predict urban parking occupancy using the LSTM model focusing on challenges and limitations of collected data. It introduced some limitations of traditional prediction systems, which is low accuracy, high computational cost, need for a big amount of data or surveys, different time, weather, etc. The LSTM model also has shortcomings, which are a rapidly increasing amount of training time and model complexity as data increases. Then the description of the data collection method,information about Ingolstadt town and the structure of parking data is provided. The data processing includes dropping outliers, using interpolation to fill missing values in some time intervals, downsampling, feature engineering of time intervals, scaling free spots and max available slots in [0, 1] range,splitting data for training, validating and testing by 0.6:0.2:0.2 ratio and finally using 2 layered LSTM method to train the data. The result is calculated with and without considering event data, and obviously the former gives a more accurate result. Since longer data sequences take more time to calculate, the training was provided for 20 epochs upon the whole dataset. Around 10-15% was achieved because of the MSE after 20 epochs of training in different time intervals. Also sensitivity analysis of different events and weather changes with different parking lots is also shown.

The paper Predicting Free Parking Slots via Deep Learning in Short-Mid Terms Explaining Temporal Impact of Features[15] does the same predicting work, but works for short and mid time interval, and there's no need to have all the data of all the garages like the previous paper. The paper proposes an unique solution for people to predict the approximate amount of available parking spaces in off-street parking places for different locations and time horizons, using deep learning models based on convolutional bidirectional LSTM networks and various features related to parking data, weather, and traffic. It aims for both short-term which means from 15 minutes to 60 minutes time in advance and mid-term means 24 hours beforehand predictions with high accuracy and low computational cost. The data used for the prediction models are collected from the Snap4City platform, which provides real-time data about parking occupancy, weather conditions, and traffic flow in the city of Florence. The data are collected every 15 minutes from March 1, 2022, to June 5, 2022, for three different vehicle parking areas: a suburb hospital, a central parking, and a historical center parking. The data are pre-processed, normalized, and imputed to deal with missing values and noise. The paper compares the performance of different models with CNN-BI-LSTM model such as RNN, BRANN, CNN-LSTM, CNN-GRU, CNN-BI-LSTM and also compares the results of the with some of the solutions that are considered as state-of-the-art solutions in terms of error metrics such as MAE, MSE, RMSE, and MASE based on some free parking slots, that are independent for the parking allocation. The comparison concludes that CNN-BI-LSTM model provides the best deal between precision and computational performance.

The paper "IoT Based Sensor Enabled Vehicle Parking System" [22] develops an advanced parking management solution to solve parking difficulties and block traffic in urban environments. The study discusses how the Internet of Things (IoT) and sensor technology can be integrated together for a smart parking system that dis-

plays real time parking availability information. This system is to provide efficient parking management to help users find the available parking spaces quickly, shorten the time of searching the parking spots, and lower the environmental footprint due to traffic.
The hardware components include infrared sensors, RFID Tags, and Arduino based microcontrollers that work together to send user parking status updates. Vehicle presence is detected by infrared sensors at each parking slot and vehicles are automatically identified and billed using RFID tags attached to vehicles. The ESP8266 Wi-Fi module receives data from the sensors, then transmits the data to an IoT platform that updates a webpage with parking slot availability. With this, drivers can check whether any slots are open from a mobile device or computer before getting to the parking location.
The system operates through a four-module process: The services provided by the system are as follows: (1) Intimation of Parking Slot: Alerting users to available parking slots; (2) Parking Detection: Sensors monitor and report the slot status; (3) Billing: Based on time duration, parking fees are auto calculated and auto communicated to users via SMS; and (4) Notification: Real time updates to the users on slot availability, billing, etc.
In addition to that, this smart parking solution includes a few additional features that make the user experience better and operationally more efficient. For example, LED indicators on parking slots change their color depending on occupancy status, a real time clock module helps in precise calculation of billing duration. With IoT technology at the core of the system, it is possible and manageable remotely, in line with smart city objectives around improving urban mobility and reducing congestion. The system allows users to reserve and monitor parking spaces through an IoT interface, with little manual intervention, short wait times and space utilization optimization, particularly in the densely populated areas.
Analysis of the results shows high precision in vehicle detection; slot occupancy is reliably identified to within 0.4 meters. The accuracy and efficient communication of data to users aid in the scalability of the system to large scale urban applications. The authors also talk about the environmental benefits of this solution, in that reduced parking search times reduce vehicle emissions and improve air quality.
Finally, the paper shows that an IoT based smart parking system can help parking management in urban areas, reducing the urban congestion in an eco friendly manner. According to the authors, this system should be implemented in high traffic areas like shopping districts, and that further improvements could include additional payment options and adding IoT capabilities for future urban infrastructure needs.

In the paper titled "Real Time Car Parking Detection with Deep Learning in Different Lighting Scenarios" [23], Yusuf and Mangoud tackle a common problem, as it becomes more and more difficult to find a free parking spot, especially in urban areas where congestion of vehicles is common. In this work they propose a camera based intelligent parking system that utilizes deep learning and image processing to provide better parking slot detection for different lighting conditions. The researchers began by developing a simple parking detection system based on image processing which determined whether parking was available by counting how many white pixels there were in each parking slot image. But in low light environments

this proved insufficient, and a more sophisticated system was then developed.
To deal with the constraint of traditional image processing, this study employs YOLO (You Only Look Once) deep learning model, including YOLOv3 and YOLOv8 versions, to conduct precise and real time object detection at various conditions. Although it is well known for its speed and accuracy, the YOLO model, brainchild of the respected You Only Look Once, processes images by sorting them into grid cells, which are then used to predict, based on the boundaries in parking slots, whether a vehicle is present or not. Webcams are strategically placed to capture real time images of parking areas and their availability and occupation can be updated continuously. In the presence of these upgrades, the proposed system does well in varying lighting conditions and is able to separate occupied and vacant spaces at night, with a high recall rate and accuracy.
Additionally, the paper explores application of the system in large parking areas, where YOLOv8 outperformed by up to 97.5% precision and 92.5% recall. The researchers refined the algorithm to work in large parking lots where lighting and shadows and varying camera angles can make detection accuracy difficult — a feat made possible by the model's powerful detection capabilities. The system was tested for both small and large parking environments under low light conditions, testing the robustness of the system beyond that of traditional image based methods.
The final remarks made by Yusuf and Mangoud indicate that parking space detection and usability can be very much improved using a deep learning approach, especially with YOLOv8. The study offers to improve the model with recordings of various training sets to mitigate the effects of lighting variability and complicated environmental factors. This smart parking system that uses deep learning and real time video analysis offers promise to the urban parking challenges with a potential to reduce traffic and reduce environmental impact by reducing vehicle emissions from lengthy parking searches.

In their latter paper titled "Design, System and Fabrication of IoT Based Car Parking System" [32], Singh described challenges faced with urban congestion and inefficiencies in the current parking system by designing an IoT enabled multi story parking system. The aim of this study is to improve parking space allocation for reducing waiting time and enhancing the convenience of users in densely populated areas. As we find that traditional parking systems suffer from limited accessibility, high congestion rates, and better space utilization, they propose to identify the significant issues of such systems. The proposed IoT based proposal consisting of several sensors, communication modules like Arduino and IR sensor to measure the number of occupants of several floors are available live time.
In order to make the system practical, the authors designed and tested a scaled down prototype with an automated lift mechanism powered by a DC motor. This is a lift which enables the vehicle movement among floors, corresponding to the actual multi level parking. Individual parking slots are sensed using the IR sensors which relay the data to a central interface. By showing this information to users at the parking entrance, users get a chance to locate empty slots easily. Moreover, the integration of IoT gives an edge to take control of the IoT enabled devices remotely and provides the advantage of automated monitoring and management thereby enabling the process of working to be better promoted and also diminishes

the need for human intervention.
One notable feature of this study is to apply real time data analysis in order to reallocate space dynamically in the parking structure. The authors show that their prototype effectively reduces congestion within the parking facility while enhancing user satisfaction and facility operational efficiency. The use of IoT technology helps in making cost effective monitoring and control of parking systems by addressing both economical and environmental problems that occur in the high urban traffic.
Overall, Singh et al. show that there are substantial improvements to parking management using IoT based solutions in multi story facilities. The proposed system not only alleviates congestion while minimizing space utilization but also implements automated vehicle movement and real time occupancy tracking. The results of this study suggest that IoT offers great opportunities to transform these traditional parking structures into smart, efficient and user friendly structures, and a promising avenue for future urban planning.

In the paper titled "CMCA-YOLO: A Study on a Real Time Object Detection Model for Parking Lot Surveillance Imagery" [39] Zhao et al. have attempted a new solution to the complexities of real time object detection in parking lot environments especially pertains to small sized or overlapping objects like pedestrians, parked vehicles etc. To enhance detection accuracy and processing efficiency, the authors introduce the CMCA-YOLO model, which leverages two key components: Criss-Cross Attention (CCA) and Multi-Spectral Channel Attention (MSCA) mechanisms. Together, these mechanisms enhance the model's capacity to discern overlapping targets and to adapt to variations in environment encountered with parking surveillance (i.e. lighting and perspective) in the scene.
Built upon the YOLO (You Only Look Once) framework, CMCA-YOLO model is a modified version of YOLO that optimizes it to identify objects in complex scenarios common in a parking lot. The model is given access to the CCA mechanism so it can collect pixel level contextual information to help it differentiate features across horizontal and vertical paths, which is especially relevant to crowded and occluded scenes. This further refines this approach by using the MSCA mechanism to analyze and compress channel information based on frequency components so that the model is able to focus on important details even under varying lighting or environmental conditions.
On the custom dataset containing 4502 parking lot images, the CMCA-YOLO model is tested with an mAP@0.5 of 0.895, and outperforms several benchmark models in terms of detection precision and speed. Compared with other state of the art models, such as YOLOv5 and SSD, the CMCA-YOLO was shown to be excellent in detecting small and overlapping objects while maintaining a high frame rate, making it ideal for real-time applications. It is demonstrated how the model is robustly adaptable to such use in intelligent parking systems for maximizing efficiency and minimizing safety risks in the urban environment.

Safran et al. address the problem of vehicle identification and monitoring within smart parking systems in the study 'Efficient Multistage License Plate Detection and Recognition Using YOLOv8 and CNN.'[29] Given the high installation costs

and low functionality of traditional sensor based systems, the authors propose a multistage deep learning approach using computer vision. To solve this problem, they combine YOLOv5 for license plate detection, YOLOv8 for character recognition, and a custom CNN model for character recognition, tuned for the Saudi license plate format. Using existing surveillance infrastructure, this setup reduces cost and improves accuracy and functionality.
To identify vehicles in real time, the authors designed a multistage pipeline that runs over images from parking lot cameras, efficiently detecting and recognizing license plate numbers. A custom dataset of Saudi license plates is used to achieve recognition accuracy of 96.1%, compared to 83.9% for single stage methods. The capabilities of YOLO models for segmentation and the custom CNN trained to distinguish overlapping characters under variable lighting conditions enable this advancement. Integration of the system into smart city infrastructure allows for real-time updates on parking lot occupancy and vehicle movement through the system's web based dashboard.
Through experimental deployment, Safran et al. show the system can effectively improve parking efficiency, decrease congestion, and contribute to sustainable urban planning. This study has important contributions for computer vision applications in smart parking, particularly in regions that need dedicated license plate recognition.

In the study, Evaluating the Performance of Ensembled YOLOv8 Variants in Smart Parking Applications for Vehicle Detection and License Plate Recognition under Varying Lighting Conditions [33], Singh et al. present multiple YOLOv8 variants to enhance vehicle detection and license plate recognition in smart parking applications. To overcome the drawbacks of single model approaches, we study an ensemble of YOLOv8 models, which consist of nano, small, medium, and large models, each evaluated for precision, recall, mAP (mean average precision), and inference time under varying lighting conditions. Smart parking management is balanced with real time processing needs to provide more accurate information.
The methodology trains each YOLOv8 model on vehicle and license plate datasets and evaluates each of them using TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution). With this multi criteria decision making technique we identify the optimal model combination based on the performance metrics that are important to smart parking, i.e., detection accuracy and processing speed. The TOPSIS analysis highlights two top-performing combinations: (1) YOLOv8 small for vehicle detection and YOLOv8 medium for license plate recognition, and (2) YOLOv8 medium for both tasks at different light intensity from bright daylight to low light.
The study further improves license plate recognition by implementing pre-processing techniques such as luminosity modulation and DPI enhancements which significantly improve OCR accuracy, especially under difficult lighting conditions. The system is tested under controlled lighting conditions by the authors and shown to be robust to a wide range of conditions, with high detection accuracy and efficiency.
In this study we prove that ensembled YOLOv8 models are a viable approach for using models in real time smart parking applications, and that they can be scaled to accommodate fluctuating lighting conditions of urban environments. Finally, the

authors propose for YOLOv8 ensembling to be used as a cost effective and efficient alternative to existing systems in the smart parking context, and for their integration into wider urban traffic management.

The paper "A New Approach to Detecting Free Parking Spaces Based on YOLOv6 and Keyframe Detection with Video Analytics" [33] addresses a new approach to detect free parking spaces using YOLOv6 and keyframe detection with video analytics to solve parking problem. The authors also demonstrate that the need for good parking management to eliminate traffic congestion and improve safety increases as urbanization and car usage grow. The paper proposes an automated approach based on YOLOv6 and the use of keyframe detection to automate the CCTV monitoring of the parking areas as CCTV is used manually to monitor the parking areas.
To reduce the computational cost, our proposed method aims at reducing the number of learned parameters by reducing the redundancy in repetitive frames of the video. Keyframe detection technique is used to extract meaningful frames from the video to optimize memory usage and to enhance processing efficiency. The videos from open parking lots (the DLP dataset) were taken for the experiments in monitoring the parking 24/7. The author sampled a video with 11,309 frames every 20th frame and the obtained 95% rate of compression. The presented method reduces the computational cost while maintaining the accuracy of parking space detection by a factor of 1000.
Using the DLP dataset as input, we trained the YOLOv6 deep neural network over keyframes with mean average precision (mAP) of 96.6% in detecting empty parking spaces. Compared to previous studies using such versions as YOLOv5 and YOLOv7, their detection accuracy varies based on how they are implemented. The paper finally compares the proposed YOLOv6 based approach with other methods by showing that it offers competitive performance in terms of precision as well as overall effectiveness.
The authors also investigated methods that can help to improve learning ability, like cropping, flipping and rotating. Transfer learning was also used to strengthen the model. To demonstrate the effectiveness of the proposed approach, it was tested on individual frames and videos, and the results were in high accuracy on empty parking space detection.
This study provides a significant contribution to smart campus or smart urban regions high capacity parking area management in the context of the smart city infrastructure. But this method works differently from those traditional systems, as it helps to maintain parking spaces available efficiently by minimizing human intervention and reducing traffic congestion and improving resource management. The study also suggests that this could be further improved to include modules to notify drivers when they are improperly parking, for instance, or mapping layers to direct drivers to empty parking spots for greater practical utility.
The proposed method is generally an innovative application of video analytics to the monitoring of parking areas.

The research paper 'Vacant Parking Guidance for Open Parking Area through Image Processing Technique' [35] by Suhana Sulaiman et al. aims to solve the problems

that drivers face when looking for parking in open parking areas in urban environments having high vehicle density. To this end, the authors propose an image processing based approach to direct drivers to the open parking slots, usually found in commercial complexes and public places. The purpose of this work is to offer an efficient and environment friendly solution to parking management by minimizing the time and efforts required to search for empty spaces, and hence minimize carbon emissions.
It proposes an image processing based solution to determine the availability of parking slots in an open parking lot. Methods such as segmentation, enhancement, and comparison of images of parking areas are used to determine vacant and occupied parking slots. Symbols like green 'O' for available slots and red 'X' for occupied ones are placed on the processed images, to help drivers find available spaces. Then, the images are shown through a guidance information system which can help drivers find parking slots without needlessly driving.
Aerial images of open parking lots were analyzed using MATLAB software to implement the system. To identify occupied and vacant slots, the researchers used a set of captured images to create a reference background to compare against, which was done by pixel wise comparison. Next, segmentation and enhanced processes were used to highlight parking slots, enabling accurate detection of these that are open. The results of this study show the efficacy of the proposed image processing technique for identifying parking availability in open lots. In the second section, the authors point out that as the number of parking slots increases, the adjustment of marker sizes and fonts is needed to maintain visibility. It also concludes that a software based approach is not only economically feasible, but also easier to implement than the hardware based with sensors. The researchers propose that future work might aim to improve the robustness of the system across different environmental conditions as well as add features like notifications to drivers.
The contribution of this paper to the development of smart parking solutions for open areas is through the use of image processing techniques for parking management. It complements the growing body of literature on parking detection methods ranging from sensors to machine learning, by presenting an alternative that is low cost and adaptable.

In a paper titled 'Parking Slot Identification System using SVM Classifier and HOG Features' [34] by D. Sivaganesan et al, a novel and cheap method to identify vacant parking slots is presented based on 'Support Vector Machine (SVM) classifier and 'Histogram of Oriented Gradients (HOG) feature extraction. The main incentive for doing this study is to produce a parking identification system that is both efficient and affordable enough to be implemented in urban and rural areas.
Thus, the authors point out the imperfections of present parking systems, most significantly in terms of the high costs of installation and maintenance of sensory solutions. Such systems, however, are too expensive to even be considered by rural regions. In order to counter this, the proposed method develops a low cost but effective parking management system to identify free parking slots of accurate 94%. Visual information of images is extracted using HOG features and an SVM classifier to classify whether a parking slot is occupied or not.
The proposed methodology utilizes HOG features to capture the gradient informa-

tion of parking slot images, which is able to make the approach robust to changes in the lighting condition and minor distortions. Each parking slot is then classified as being "occupied or 'free' using the extracted HOG features as input to the SVM classifier. The authors experimented with the model on a dataset of parking slot images using GridSearchCV to optimize hyper parameter tuning. The study results indicate that combining HOG features with the SVM classifier leads to a high level of parking slots identification.
This paper discusses why and how SVM is the best choice to use for this particular application, rather than another machine learning algorithm like neural network. For smaller datasets, SVM was found to be more efficient, and better at handling overfitting issues. Additionally, the system is robust against structured objects such as automobiles, since both HOG features are used for object detection.
The study presents an alternative to traditional sensor based systems which is more economical and accessible and a comparison between different parking slot identification techniques. This could be a very attractive cost effective method especially in rural areas where there is a budget and infrastructure limitation in advanced parking management systems.
Finally, the paper presents an effective parking slot identification method utilizing the strengths of SVM and HOG to perform with a high accuracy under a low cost. In addition to improving efficiency by utilizing the parking space better, this approach also allows smart parking technologies to reach a larger demographic – the poor in underserved rural areas, among others. The authors then write that future work could delve deeper into making the system more convenient and reliable by employing more features, such as real time updates and notifications to users.

This paper titled "Deep Learning-Based Parking Occupancy Detection Framework Using ResNet and VGG-16" [36] presents a parking occupancy detection with the use of cameras in place of sensors since it is more economical to implement. This system responds to traffic problems such as traffic congestion through deep learning features in the detection of open parking spots within the string environment with CCTV cameras in both indoor and outdoor settings.
The study primarily employs two convolutional neural network (CNN) architectures: To improve the efficiency of ResNet50 a hybrid model with support vector machine was used whereas for optimizing VGG16 an incorporation with OpenCV was done. All these models are trained and their performances tested on the PKLot database which was captured when the parking lot environment was lighting and in different weather conditions. Good results were obtained with ResNet50, a deep residual network with good feature extraction and less vanishing gradient problem; it was 98.9%. Again, the VGG16 model with sequential layers of smaller kernel sizes, though had its constraints in different conditions, recorded 93.4% accuracy.
In their methods, the authors used the following steps of image pre-processing and feature extraction. For occupancy status recognition, features are obtained in the same manner as in step 1 from ResNet50, and result in classification by using an SVM. The purpose of the SVM classifier is to find the best hyperplane, which will provide the greatest distance from the points corresponding to occupied and free positions. For VGG16, the common directionality of the marking for parking lanes are in horizontal lines and hence the next step isolates on detecting horizontal lines

by applying the Canny edge detection and Hough transformation to perform exact boundary marking.
It is established that CNN-based models can auto-detect parking space effectively and accurately thus improving the management of traffic within city centers and achieving environmentally friendly goals. These concepts can be an integrated part of smart city strategies, decreasing the infrastructure expenses, and providing the real-time park assist mode. Thus, recommendations for subsequent studies are made in terms of enhancing precision in cases of changes in weather and lighting conditions.

The paper titled "PSDNet: "A Breakthrough Parking Space Detection Network Powered by YOLOv8" [30] deals with the problem of parking occupancy detection to mitigate the problem of congestion in urban areas. Built on the YOLOv8 concept, PSDNet has the ability to accurately capture open parking spaces with a high degree of efficiency. YOLOv8 introduces three core architectural blocks: Backbone, Neck, and Head. The Backbone uses modified CSPDarknet53 where cross-stage partial connections enhance feature extraction across the layers. The Neck, based on Feature Pyramid Networks (PAN-FPN), fuses several levels of feature maps for jointly improving the semantic feature and spatial localization. To increase accuracy in other circumstances and to make classification and detection independent of each other, the Head section uses a decoupling mechanism.
The researchers employed YOLOv8 in the Parking Space Availability dataset and realized an F1-score of 98.7 percent, which is significant than the earlier models, YOLOv5 and YOLOv7. This paper demonstrates how PSDNet can be applied to parking management as a real-time, machine-learning, computer-vision-based solution to improving parking in urban environments. The PSDNet system avoids the need for the installation of sensors, thus being cheaper and scalable for smart cities through the use of a camera-based system. Future can add more features like IoT to alert the drivers about availability of the slots and make the system more useful.

In the paper titled "Shine: A Deep Learning-Based Accessible Parking Management System," [26] the authors introduce an innovative parking management solution focused on preventing misuse of accessible parking spaces in South Korea. The "Shine" system employs deep learning techniques, particularly the YOLOv7 object detection algorithm, to identify vehicles, license plates, and accessible parking badges in real-time. This approach enhances the accuracy and efficiency of traditional license plate recognition (LPR) systems, which often face challenges under varying lighting, weather conditions, and noise interference. Unlike conventional LPR systems, Shine incorporates a transfer learning-based model trained with a specialized dataset, including images captured in various environmental conditions, to improve accuracy in real-world scenarios.
The Shine system comprises multiple components, including a display monitor, Nvidia GPU-based mini-computer, IP camera, speaker, and LED indicators, all of which work together to verify if a vehicle is eligible to use an accessible parking space. When a vehicle approaches the accessible parking area, the system scans for an access badge and license plate number. If both are present, it cross-references

them with a central database to confirm authorization. Upon verification, a green LED and a welcome message indicate valid access, while unauthorized users trigger a red LED and warning message, helping curb parking violations effectively. The results from various object detection models reveal that YOLOv7 outperforms alternatives like YOLOv5s and Faster R-CNN in both accuracy and speed, making it suitable for this task.
The Shine system's adaptability means it can be applied beyond South Korea by adjusting its object recognition model to detect local license plates and badges. Future research aims to include features that provide real-time parking availability updates, further enhancing parking management efficiency. The study underscores Shine's potential to contribute significantly to smart city initiatives, offering a scalable and cost-effective solution to manage accessible parking spaces across urban environments.

In the paper titled "Integration of YOLOv5 Algorithm and OpenCV in Innovative Smart Parking Management Approach," [24] the authors discuss the following problem of growing popularity in Indonesia's large cities: the need for effective parking services. The four areas of the study include applying the YOLOv5 object detection model and integrating OpenCV to develop a smart parking management system. The study was conducted observing various conditions (empty, partially occupied, and full) at the BRIN Bandung office to assess the system's efficiency of identifying the parking lots and vehicles in real-time conditions.
YOLOv5 has boosted features which are Backbone, Neck and Head layers for feature extraction and object detection and therefore can be used for detecting vehicles in parking lots. Therefore, the system incorporates an efficient image processing mechanism to perform visual markings of parking spaces and available spots computation using OpenCV. The model is trained with high degrees of accuracy to enable identification of various vehicles within different backgrounds of parking lots. The findings suggest that this integrated system can efficiently recognize vehicles and notify available spaces thereby augmenting time for users to find parking, and decrease traffic. In conclusion, the authors state that the proposed and implemented system based on YOLOv5 and OpenCV is a prospective and efficient solution that contributes duly to the development of adaptive parking management. Future applications may use such an approach in other areas that comprise high human traffic, thus improving the ergonomics and efficiency.

The paper "Design and Implementation of RFID-Driven Parking Management Systems" [25] discusses the utilization of RFID based smart parking systems as an effective solution for some of the problems that are characteristic of the conventional parking systems in urban areas. This RFID-centered system employs tags on the vehicle and readers in parking structures to perform vehicle identification and access management with minimal contact. With the use of the RFID tag, the system enables drivers to enter and exit the parking areas without the intervention of man and tends to reduce traffic congestion as well as parking complications.
It applies RFID tags associated with the users' payment accounts thereby avoiding the use of money or credit cards to make the payments. This is crucial from user

perspective where such contactless methods tend to improve the overall parking experience as well as convenience, more so in the densely populated urban cities where professionally managed access control and efficient payment processing is vital. In addition, the RFID system also involves real time parking space information such that the driver can be notified through his/her car radio when the next available space is ready.
The research establishes that the application of RFID technology in parking management systems is efficient in transforming the parking solutions industry, managing the environmental impact, and improving customer satisfaction. The authors pinpoint an ability to scale and suggest that this approach can easily accommodate pressure changes that occur in developing urban centers. This work proposes RFID technology as a viable solution for the Intelligent Parking System and calls for permanent research and integration of better parking systems in densely populated cities.

In the paper "PAINet: Toward Fast and Efficient Parking Lot Lane Detection," [21] the authors propose an advanced model designed for accurate and real-time lane detection in parking lots to support automated driving. The model, named Point Angle Instance Network (PAINet), employs a unique approach by using key point clustering with embedded angle information. The author's method improves the stability of detections and training time, thus overcoming obstacles peculiar to parking zones such as occlusions, lighting, and signs.
To support PAINet, the authors have developed a dataset named ParkLane containing 16004 images with different parking lot scenes and conditions such as low illumination and occlusions. In this work, the architecture of PAINet is presented, which includes RepVGG and Hourglass network structures to achieve multi-scale feature extraction and integrate key context information. According to the above results, the PAINet achieves 85.48% F1 score and runs at 164 FPS, which is higher than many existing methods, thus, it can be utilized for real-time autonomous parking applications. The further studies and developments include expanding the proposed model for more parking-related tasks and enlarging the domain of the dataset.

The paper on "An Enhanced Car Parking Detection Using YOLOv8 Deep Learning Capabilities" [31] propose a strong approach to the management of car parking in cities through the use of YOLOv8 object detection algorithm. Admitting the difficulties in parking spaces especially in congested areas, the authors seek to increase the detection rate for parking systems. The system effectively determines parking slots are occupied by cars or not with real time in real-time using YOLOv8 which has a single shot detector mechanism to speed up the entire process of managing urban mobility effectively.
This includes data capturing by images of parking lots followed by annotation to label cars, and any available space. This structured data helps more efficient training of the YOLOv8 model, using anchor boxes to obtain higher accuracy of object identification irrespective of their shape and position. Performance in simulation is evident, with the system yielding about 98.7% accuracy, which is higher compared to previous techniques such as Hough Transform and Haar-Cascade that struggle

with complicated angles and lighting. Additional evaluation metrics of precision-recall amd F1 score curves also confirm how YOLOv8 is capable of minimizing false positive detections while optimizing detection accuracy depending on the tested scenarios. This system provides useful information for city design, providing a feasible solution to the lack of parking space, especially in rapidly developing cities.

The paper titled "A LiDAR-Based Parking Slots Detection System" [38] presents a parking slots detection system employing 3D LiDAR technology for improving parking space identification in highly populated cities. Traditional contactless systems using ultrasonic and/or camera have limitations in distance, accuracy and environmental variations in sensing. In order to resolve these problems, the authors present a LiDAR-based method that can identify free parking spaces at long distance with higher precision and reliability.
The system is composed of several modules, perception, parking lot construction, tracking and obstacle detection, and it is able to constantly detect the parking spaces. The perception module uses a LiDAR-based 3D detection network called "centerpillar" to detect nearby vehicles and assemble parking spaces by eliminating obstacles. The system also consists of a tracking system to control parking space availability without the need for extra computation.
Proven in an above-ground parking lot in China, the proposed system was more accurate in parking space detection than the ultrasonic approach. The system was evaluated to have recall rate of 93.6% and a precision of 95.2% for detecting the available slots for general or vertical parking and inclined parking and horizontal parking as well. Although there are some problems with occlusion in inclined slots, the LiDAR-based approach outperformed the benchmark methods and set the foundation for application of automatic parking in certain cities.

In the paper with the title "Research on Parking Space Detection and Prediction Model Based on CNN-LSTM," [37] the authors put forward a new model which integrates CNN and LSTM networks to optimize the detection and forecasting of parking spaces in urban areas that suffer from insufficient parking lots. As a result of focusing on the scarce resource of parking space, the model employs both past and real-time data for the estimation of the parking space utilization ratio to minimize traffic congestion and enhance the efficiency of the parking lot.
The study starts with employing CNN for spatial feature extraction with a concentration on images of parking lots. Subsequently, LSTM is utilized to address time series data in an attempt to capture temporal relationships of parking demands. By graying, edge detection and selecting region of interest (ROI) the researchers form(image pre-processing) images that improve CNN's ability of differentiating between occupied and vacant regions. Also, the Canny edge detection algorithm is employed to detect the parking lot's edges, and then the Hough Transform for line detection to have a proper division of spaces in the lot.
The results of the experiment show that the proposed CNN-LSTM model provides higher accuracy than traditional statistical methods and most of the machine learning algorithms, since it is able to analyze both spatial and temporal features of the parking demand. The model was thus proved to have better predictive capability,

with lesser errors (mean absolute error of 13.3 on weekdays and 12.6 on the weekends) than other techniques. This study shows that using CNN and LSTM can improve parking management in urban areas; future studies can build on modifying these models from time to time to adapt them to different conditions in the environment.

# Chapter 3

# Methodology

This chapter describes how the research was carried out. Starting from the information we collected on different models applied, we explored the general outline of the research process.

## 3.1 Overview of the Research Methodology

This section give a detailed insight about the research methodology and the workflow of this research. The first objective of this work is to build a highly accurate system for vehicle detection and free parking slot identification within garage environments capable of real time implementation. In order to achieve this successfully, the research process has been rigidly structured into several systematic phases thereby guaranteeing a methodical and optimal manner of developing, evaluating and ultimately deploying the model.
We started off with an extensive literature review of various research paper to understand the current state of the art of vehicle detection and parking slot identification, which specifically provides the foundational knowledge to choose appropriate machine learning models and techniques most suited to address challenges presented by existing solutions.
The dataset acquisition and preprocessing phase follows the literature review and involves acquisition of relevant data and coordinations of it for model training. The dataset is organized, data augmentation is performed to diversify the dataset, and the accuracy of annotations is ensured through very careful validation checks. But the end product of these steps is building a robust enough dataset to feed into our chosen models.
However, before using that, the model selection and training phase comes next where object detection frameworks such as Faster R-CNN, MobileNet SSD, YOLO have been assessed in order to select the most suitable model for the task. Applying each of the models, we train each with a preprocessed data set aiming to fine tune at factors that enhance the detection of the models and computational efficiency. After all the models are trained, they are all put through rigor, testing and validation to check how each of them will do under different situations.
In the next stage of this research, a balanced model is chosen for performance improvement. After conducting various tests and validation, the optimal settings for modification is chosen. Analysis of outputs by comparing them with models that trained and tested beforehand is conducted to verify the increased reliability of the

modified version of the model.
The research then concludes with an evaluation and analysis phase in which the performance of the implemented system is critically assessed using predefined metrics including precision and recall. Strengths and weaknesses of the chosen approach are compared to alternative models and insights to further improve are given.
The iterative testing and refinement help throughout the research workflow so each phase adds properly to the overarching goal of improved parking management systems. This study aims to provide a reliable, efficient and a scalable vehicle detection and parking slot identification solution for varied garage environments by systematically treating each component of the workflow.

## 3.2 Research Workflow

In this section, we added the flow chart of the whole research we conducted. This is divided into two parts. The first part shows the flow of training, testing, and analyzing result of various models, the second part shows how we improved the YOLO model's precision by targetting the empty slots.

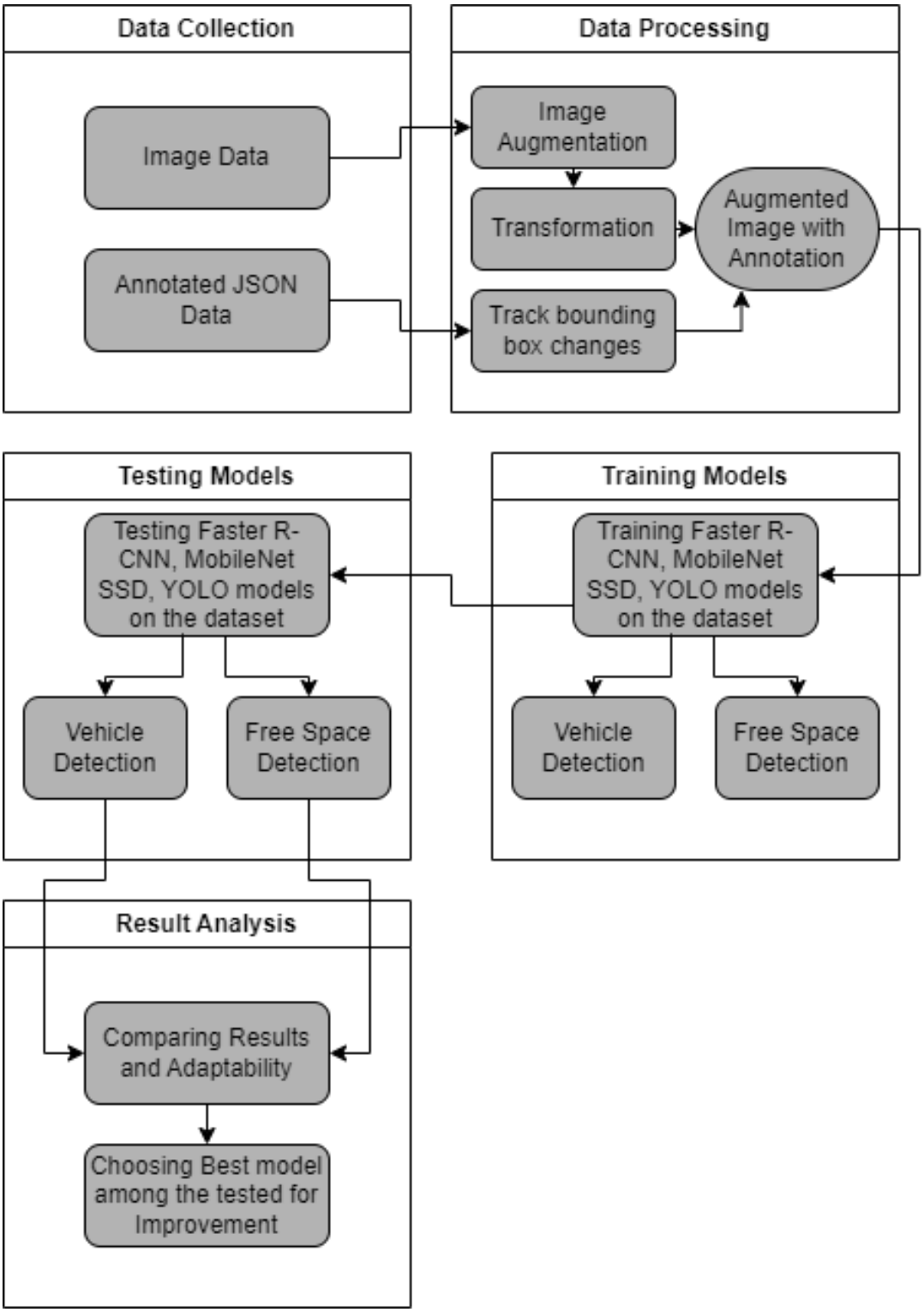



Figure 3.1: Flow chart of training different models and comparing the results.

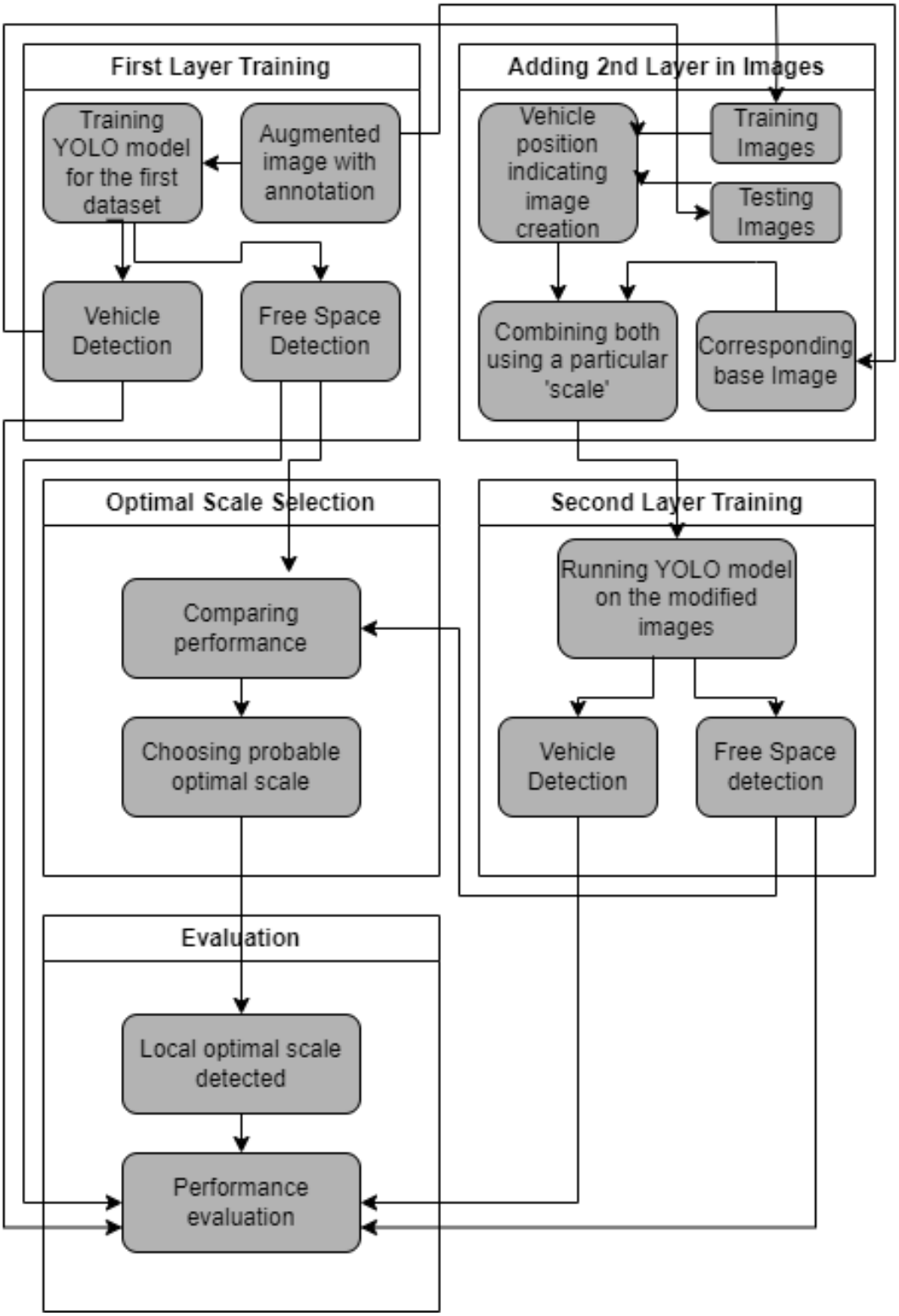



Figure 3.2: Flow chart of modifying YOLO model for better precision.

## 3.3 Work Plan

The whole process is divided into two main part- first is training and testing different models, and second part is improving the best performing model among them. For improving part, we tried to add more information in each of the image so that if we take a $n \times m$ box of empty slot from any image, we can find some information of relative position of the nearest car and the scale of the image .To accomplish all those tasks, we followed the following procedure in order.

1. Collect dataset, rearrange the image files, annotation files in separate place and arrange train and test folders separately.

2. Apply augmentation technique to increase the number of images for both train and test folders. Modify the COCO annotation file based on the changes applied on all images during the augmentation.

3. Use the augmented training images to train faster RCNN model. Keep track of `category_id` of the annotations, which has 4 values, 1, 2, 3 and 4. Where 1, 4 is for empty slots and 2, 3 for cars. Merge them and train the model using python libraries.

4. Test the Faster RCNN model using the images from augmented test folder, calculate the precision, recoil and accuracy for both cars and empty slots separately.

5. Draw the bounding boxes for both car and empty slots detected from the testing process and save them in a folder for future comparison.

6. Repeat the process 3 to 5 for MobileNet SSD and YOLO separately. We will use YOLO version 8 in this case.

7. Pick compare the results and pick the best model among those which gives higher performance and suitable for real time detection. In our case, it was YOLO.

8. Using the annotation file, list out midpoint of all the boxes containing vehicles in all the images. Then for each of the images, create a image, which can indicate the cars and the surrounding places. How we achieved this is explained in the next section.

9. After generating the car position image for all the images, we combined both images, prioritizing the original image, and replacing $2 \times 2$ pixel of the corresponding car position image pixels, separated by `scale` pixels by both vertically and horizontally repeatedly. Here `scale` is a value, which will be tested to find the optimal value of this variable.

10. Next, we use the previously saved midpoint position of car bounding boxes found from the tested images and apply the same modification on them. The difference of test image modification from train is, we use the midpoint of car bounding boxes from the annotation file for train, but we need to use the trained model to find the midpoint of car bounding box of the test images. So, a model with high accuracy in vehicle detection was needed to achieved this, which YOLO did perfectly.

11. Now, we create a different model instance of YOLO v8 and train it with the modified training images.

12. Using the modified testing images, we test the images again, and note down the accuracy, precision and recoil.

13. Then we compare the result from the second model with the result of the original model, and record the precision.

14. Repeat the process 9 to 13 by changing the value of the `scale` variable. Plot it down using `precision` vs `scale` and find a probable optimal value of scale which gives higher precision.

15. Using process 9-13 on that scale, find the precision and compare. If it improves the precision form the original model, we achieved our goal, if not, go back to previous process.

The reason all values of `scale` will not produce desired result because there is a trade-off between extra information on car position and the original image. So we need to find the optimal. After achieving it, we can train the YOLO model for both normal and modified image and save it. Then save the image modification function, which takes the image and midpoint co-ordinates and save it. For real time detection, the model do the following in order:

- select an image for detection
- Run original model on the image
- Find car bounding boxes and save them.
- Insert the image and midpoint of the car bounding boxes in the image modification function
- Run the modified image which came out of image modification function in the modified model
- Save the Free slot bounding boxes.

This way this model can do real time detection for both car and empty slots more accurately.

## 3.4 Dataset

This section analyzes the dataset collection, verification and modifications applied on it. The used dataset in this research was taken from a secondary resource that was carefully chosen to fit well with our study of vehicle detection and free parking slot identification. The details about the dataset collection and processing is discussed in detail below.

### 3.4.1 Data Collection

This research was carried out on the dataset that is the source of this data and its collection and preprocessing is discussed in detail below. To ensure that we utilized previously used datasets, we performed a thorough search through existing projects and identified several preprocessed datasets available on RoboFlow and Kaggle. We chose two datasets among them- "Parking Slot Detection" [28] dataset and "PKLot Dataset" [3] as the training and testing dataset suited to our time limit. **We will call them the first and the second dataset respectively throughout the whole paper.** This also follows the requirement that the dataset images should be from the same garage to ensure better performance in detecting free slots.
The fist dataset dataset contains a total of 426 images and the second dataset contains 12,416, organized into three folders: train, validation, and test. The images feature outdoor car parking areas, aligning well with our primary objective of categorizing objects into three sections: vehicles, background and empty spaces. The dataset satisfies our requirements sufficiently by offering sufficient scenarios for vehicle and free space detection.
There is a limitation of the first dataset in the limited number of images when compared to standard datasets used in training deep learning models with. But this datasets helps to estimate performance when images are not available to train. But the seconod dataset has no such limitation. Some of this paucity of data may influence the generalization capability of the trained models in the first dataset. In order to counteract this problem, we opted for applying image augmentation approaches to artificially multiply the dataset. As no augmentation methods have been previously applied, the dataset can be further enhanced by applying such techniques as rotation, scaling, flipping, and transformation of colors. Doing this would increase the variations to which the model is exposed, i.e., the model would be more robust and accurate in classifying vehicles and empty slots under dynamic assumptions. We applided the same augmentation in the second dataset, but did not multiply it's size as it is already very large.
We attempt to correct the first dataset's limitation through augmentation so that the model operates better and generalizes better to real world parking cases. The addition to the dataset would have a huge effect on the effectiveness of our machine learning in being able to accurately detect vehicles as well as free parking spaces even with limitated data.

### 3.4.2 Data Analysis

The datasets consists of images of a parking space in JPG format, organized into three distinct folders: The annotation files consist of *train*, *valid*, and *test* each having its applicable annotations. To ensure uniform resizing of all images to a fixed resolution size $640 \times 640$ pixels for machine learning model processing and consistency during processing, all images has been resized to the same format. Before the model was trained, there was no data augmentation, which allowed us to quantify how performance would fare even on the raw data.
JSON files containing annotations for all images are provided as well, with annotations in the COCO (Common Objects in Context) format. Detailed metadata, including image acquisition dates, bounding boxes indicating the vehicles in each image, and the image areas bounded by these bounding boxes, are included in these


Figure 3.3: Sample images from the first dataset

Figure 3.4: Sample images from the second dataset

annotations. The data is organized separately in *train*, *valid*, and *test* classes of JSON files to use for the different models' training, validation and testing phases. This was done in order to verify manually the reliability and accuracy of the annotated data. The mapping involved aligning the bounding boxes with the respective images and then scrutinizing them for perfect fit and correct shape. Though there were minor differences in height to width ratios in some bounding boxes, these differences were minimal compared to the overall data set, and did not erode data integrity. Due to this they were not adjusted since they were considered negligible for purposes of this research.

There is the opportunity to augment the dataset further in the absence of data augmentation. One way to increase diversity in training data, and to help improve generalization of models, would be to augment the data with rotation, scaling and flip transformations. Nevertheless, initially grounding in the unaugmented dataset enables initial performance evaluation of the models in their standard conditions.

In general, this complete verification and data quality data analysis significantly guarantees that the dataset consists of good quality vehicles and free space detection and appropriately trained and evaluated for machine learning models to detect vehicles and free space in parking garages. There was some height-width proportion problem, that was insignificant compared to the total amount of data, so was

ignored.

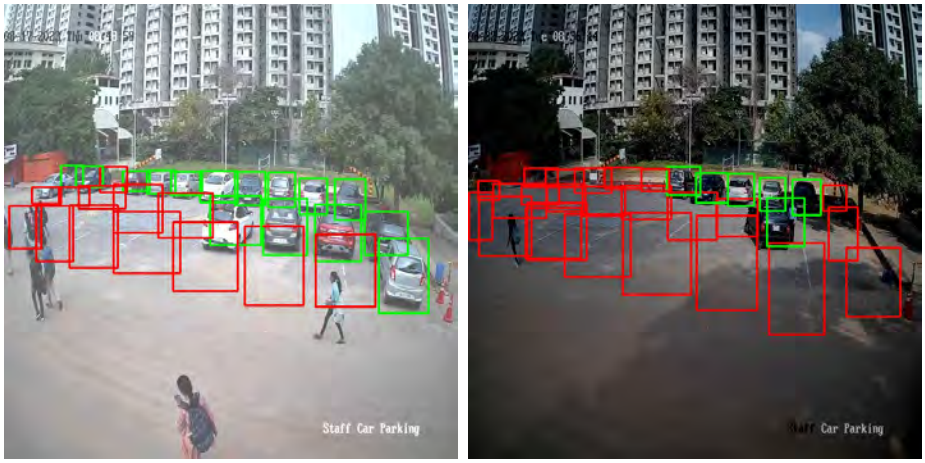



Figure 3.5: Verifying the accuracy of bounding boxes against the images. Red for empty slots, green for cars.

### 3.4.3 Data Preprocessing

To make the datasets more convenient and efficient for training and evaluation the dataset was reorganized for the data preprocessing step. The original datasets contained images, and associated JSON annotation files within the same directories. However, when accessing images programmatically, the setup presented a number of challenges, as file access commands which could inadvertently pull annotation files, were programmatically invoked.
To this we restructured the datasets such that the JSON annotation files were moved to separate directories. In particular, we took the annotation files out of the image folders so that using the `os.listdir()` command in Python would only give us the image files from those folders. By providing more error free, and straight forward access to the images during preprocessing and model training this adjustment streamlined the data loading process.
The first dataset was a bit on the limited images side, and we knew that the model could be limited in the ability to generalize, and also it could lead to overfitting, so we applied data augmentation techniques and miltilplied it's size by 5. To get broader robustness and performance for the machine learning models, the diversity and quantity of training samples of the dataset needed to be augmented for the increase. The same augmentation technique is used on the second dataset, but the dataset size was kept as the original.

#### Image Augmentation

And the most important step is data augmentation at which we can create artificially modified versions of the existing images. In contrast to such functions with common datasets such as MNIST, when we are working with object detection tasks it is crucial that the transformations we apply to the images are correctly reflected in the associated annotations as well. This ensures that the bounding boxes and the category labels of the augmented images have aligned with the visual content of it.

However, as an initial step towards achieving this, we used our Python code to load the images alongside their COCO JSON annotation files. Using the `albumentations` library, a strongly powerful and flexible augmentation library that, on the one hand, allows for easy application of more complex transformations while preserving annotation integrity.
The following augmentations were applied to the dataset, each with a specified probability, to introduce variability and simulate different real-world conditions:

- **Horizontal Flip** with a 50% probability: Just like image, this transformation will transform the image along the vertical axis, which means it will reorient the left-right. This allows the model to learn this translational invariance to horizontal pose (vehicle and parking space).
- **Brightness and Contrast Adjustment** with a 20% probability: It randomly augments the brightness and contrast level of the images. It performs multiple lighting condition simulations, for example shadows and glare, that are commonplace in outdoor environments.
- **Random 90-degree Rotation** with a 50% probability: They rotate images by 90 degrees in some random direction. Adding a degree of diversity with respect to vehicle orientations, parking slots will help model recognise objects from varied angles.
- **Resizing** to a fixed size of $1024 \times 1024$ pixels: In general, batch processing requires equal image sizes, and the models are expected to get the same input at all the time. It also ensures that all images end up fitting practically within the proposed network input sizes.

Images were augmented sequentially by these augmentations one by one, 5 times for each images for the fist dataset, and 1 times for each images for the second dataset. This approach ultimately increased the size of the first dataset by five fold, greatly increasing the volume of training data experienced. Moreover, during augmentation the transformation parameters were also applied to the bounding boxes and category IDs associated with each image. The annotations were adjusted in this meticulous way so that the annotations were still accurate, and the transformed images were to the exact parameter.
This resulted in the training image number of first dataset becoming 1,780 images long, and the testing image number of the first dataset grew to 110 images. This increase in data also introduced variations in scenarios and conditions; and considering the amount of data now available to the models, not only did they have more to learn from, but they were learning from a much wider range of conditions and scenarios. Through exposing the models to various orientations, lighting and scale variations we hoped to help the models learn to generalize to unseen data. The second dataset had 8691 for training, and 1242 for testing as the original dataset.
It is also an aspect of regularization since it helps to decrease the probability of overfitting much more data. We were able to do this by feeding the models with versions of the images that have been augmented. However, the models are fitted with the patterns and features that have the ability to generalize and transform with the varying appearance of the image.

Finally, It can be said that data preprocessing and augmentation was very important to enable the usage of the dataset. We reorganized the dataset structure to provide a solid footing for training machine learning models to adeptly detect vehicles as well as identify available free parking slots across varied conditions.

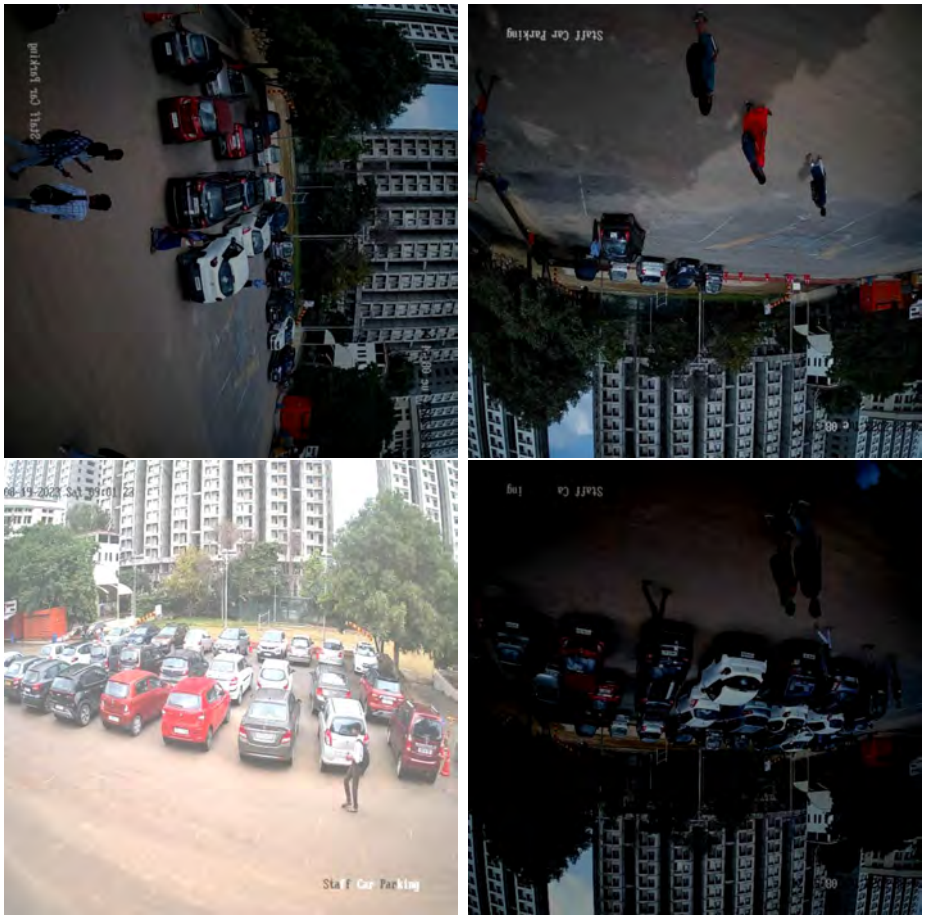


Figure 3.6: Sample images from the first dataset after augmentation

## 3.5 Model Description

Given that an effective solution for vehicle detection as well as identification of free parking slots requires choosing machine learning models that satisfy the trade off between the detection accuracy and computational complexity, developing an accurate and efficient solution is necessary. After conducting extensive research, we selected three prominent object detection models based on their specific characteristics: MobileNet SSD, YOLO, and Faster R-CNN. This section explores the details of pros and cons of those models.

### 3.5.1 Faster R-CNN

Faster R-CNN is a state of the art object detection framework which extends R-CNN and Fast R-CNN [4]. It fuses the process of region proposal generation and object detection into a single, efficient unified network [5]. The R-CNN paradigm used selective search algorithms to produce region proposals [2], and used a convolutional

neural network (CNN) to classify each region individually. Due to the redundancy in CNN computation for the regions of perpendicular overlap, this method was computationally intensive. Fast R-CNN shared convolutional computations among regions while still being dependent on region proposal methods [4]. Regional CNN (Faster R-CNN) brought the region proposal network (RPN) to the foreground, which, in contrast with traditional detectors, directly generates region proposals within the CNN architecture and provides end-to-end training and real-time object detection [5].

The advantages of applying Faster R-CNN to vehicle and free space detection in parking garages are that. First, it employs a robust feature extraction and region proposal mechanism, so it achieves accurate vehicle detection [5]. Second is the end-to-end training, where feature extraction, region proposal, and classification are circumvented to simultaneously optimize for all three in a step [5]. Finally, the model can learn on different datasets, and generalize to variations in illumination, occlusions, and perspective of CCTV footage of garages [10]. Finally, Faster R-CNN can be trained to recognize unoccupied parking areas if we have the labels for empty parking places that are not classes themselves or we have the spatial relation between the vehicle and parking slot to deduce it [10].

In this work, we use Faster R-CNN to overcome the hurdles in vehicle detection and free parking slot identification in garage environments. We annotated the images in CCTV footage from parking garages with bounding boxes for vehicle and empty parking slots, including various scenarios with diverse vehicle arrangements and lighting conditions. Using the collected dataset, a garage context specific Faster R-CNN model is trained through fine-tuning a pre-trained CNN backbone for models to be adapted to the specific features of vehicles and parking spaces in garage conditions [5]. Using robust feature extraction and region proposal mechanisms, the model's prediction is bounding boxes surrounding vehicles with good accuracy. Detecting empty parking slots also proved to be challenging, and we addressed this via defining one more empty slot class and including some contextual information about the layout of the parking area [10].

In order to be able to detect free spaces more effectively a number of additional steps, including spatial reasoning and post processing, were conducted in order to consider variations in vehicle positioning and spacing. mAP values were computed for both the vehicle and free space detection and used as the metrics for model evaluation [5]. In this application, Faster R-CNN was compared to other algorithms [10].

Although Faster R-CNN provides a great deal of benefits, it also has met several challenges. Parking arrangements vary widely so that the same amount of vehicles may occupy the same spaces in a different arrangement as well as different orientations, which affects inaccuracy of free space detection [10]. Distinct visual features for empty parking spaces are missing compared to vehicles, thus making detection of empty parking spaces harder [5]. Detection accuracy was affected by variations in lighting and occlusions of structural elements or other vehicles [10]. It also consumes a fair bit of computational power and requires designated hardware for it to perform in real time [5]. To overcome these challenges, we experimented with data augmentation techniques, thought about the annotation strategies, and optimized the model's architecture, where the accuracy balances against efficiency.

### 3.5.2 MobileNet SSD

MobileNet Single Shot MultiBox Detector (SSD) is an object detection framework that takes advantage of the MobileNet architecture's efficiency [8], while maintaining the fundamental ideas behind SSD detection methodology [6]. Depthwise separable convolutions are adopted by MobileNet to minimize computational cost of convolutional neural networks in mobile and other embedded vision systems [8]. However, SSD is an object detection algorithm that discretizes the output space of bounding boxes by default over different aspect ratios and scales over each feature map location [6]. Integrating MobileNet with SSD, MobileNet SSD takes a fair trade off between speed and accuracy [6], thus can be used in real time for running on devices with limited resources.

Several potential advantages come from applying MobileNet SSD to solve vehicle and free space detection problems in parking garages. Secondarily, because of its lightweight nature, MobileNet is well suited for real time detection in constrained processing power devices, hardware like the one embedded into a CCTV camera [8]. Secondly, SSD's one stage detection approach can be run faster than two stage detectors such as Faster R-CNN [6], which is desirable for real time monitoring. The last is that the implementation of the model is size reduced so that it can be more easily deployed and integrated into existing systems with little or no hardware changes [6].

In our research we choose MobileNet SSD to detect vehicles and free parking spots within garage environments to compare with other approaches like Faster R-CNN or YOLO. To ensure consistency of evaluation between different models, we used the same dataset of annotated CCTV images from parking garages. This dataset was used for the MobileNet SSD model training task, and adapted via fine-tuning to the specifics of the parking garage environment. The model was then used to demonstrate its ability at accurately detecting vehicles under different lighting conditions and in different orientations. We also investigated whether MobileNet SSD can recognize empty parking slots, hence requiring it to perceive spatial context, and that vehicle was not present in this area.

MobileNet SSD has its demerits, though. While depth wise separable convolutions in MobileNet are efficient, they may compromise a model's ability to learn high level features for discriminating between the presence or absence of parking spaces [8]. Finally, SSD's one stage detection method may not achieve comparable localization accuracy as two stage detectors, resulting in less precise bounding boxes around vehicles, free spaces [6]. MobileNet SSD struggles with objects of different sizes, like vehicles at different distances from the camera, because they use a less powerful architecture that can't handle scale variance well [6]. In addition, these empty parking slots have less prominent features that are harder to detect, and may be beyond the capability of MobileNet SSD to detect for classes that require high-level contextual understanding [10]. As lighting and partial occlusions by structural elements or other vehicles can also vary and MobileNet SSD is given a more lightweight nature, these factors can decrease the detection performance [6].

This research evaluated the effectiveness of MobileNet SSD as an improved comparative object detection model for vehicle and free space detection in parking garages compared to traditional schemes [10]. In terms of computational efficiency and speed, MobileNet SSD has the advantages, although observed challenges suggest that MobileNet SSD might not be the best choice for situations that require high

detection accuracy and robust feature representation [10]. This paper draws attention to which model to choose from the perspective of being efficient in terms of executing the design while maintaining the complexity of recognizing vehicles and free space in different garage environments [10]. The insight presented here helps proceed in developing effective and feasible parking management systems.

### 3.5.3 YOLO

Based upon great speed as well as accuracy, You Only Look Once (YOLO) has advanced the field of computer vision [7]. However, unlike traditional region based methods like Faster R-CNN, YOLO are regression based methods, detecting objects in the whole image in one step, predicting bounding boxes and object class probabilities rather than performing detection in two steps [7]. Inherently, YOLO is naturally such a good fit for real time detection applications because it processes images so fast [7].
YOLO divides images using a single CNN architecture and splits images on a grid of the dimensions $S \times S$ [7]. The grid cell is supposed to predict a fixed number bounding boxes, confidence scores for those boxes, and class probabilities for each [7]. YOLO frames detection as a single network problem, which avoids the need of additional region proposal stage significantly boosting the detection speed without sacrificing accuracy [7].
There are benefits to applying YOLO to vehicles and free space detection in parking garages. Its real time capability makes it good for CCTV processing for which the occupancy status of the parking can be depicted immediately [7]. Second, YOLO's global thinking approach allows training and testing on the whole image resulting in much better detection of contextual information and minimizing false positives [7]. In particular, this characteristic is particularly useful in environments like parking garages, where vehicles and vacant slots can present in all possible configurations [7].
Additionally, YOLO designs its architecture to handle multiple objects of different classes in a single image [7]. In this case it can identify vehicles as well as free parking slots at the same time [7]. In this way, YOLO uses anchor boxes and multi-scale detection layers for detecting objects in different scales and thus can detect vehicles of different sizes and distances from the camera [7].
One of the reasons that YOLO may perform better as compared to Faster R-CNN [5] and MobileNet SSD [6] in this particular application is YOLO's trade off between speed and accuracy [7]. At the same time, Faster R-CNN is accurate, but computationally expensive and the speed may not satisfy real time use cases [5]. MobileNet SSD is like efficiency, which may not be robust enough for larger domains like parking garages [6]. Thus, YOLO offers a good tradeoff in detection accuracy and inference time while maintaining a reasonable speed of 30 fps and high accuracy on RetinaNet (60fps) [7], which is good enough to deploy real world applications where speed is as important as accuracy.
In addition, its capability to generalize from natural images to novel domains means that YOLO is robust to various parking garage settings without a huge performance loss [7]. Its capabilities to learn from the whole image context and extract robust features make it suitable for situations when CCTV footage presents variations in lighting conditions, occlusions and camera angles [7].

In this research, the scope of application of YOLO is to determine which vehicles and free parking slots can be accurately detected within garage environments. Using this existing infrastructure to track the parking occupancy, we introduce a system that leverages YOLO's strength in real time detection and contextual understanding to use existing CCTV [7]. This approach helps in saving the cost of investment for additional hardware investments like IoT devices and provides economical parking management.

This understanding of the basis for YOLO's superior performance in this context points to the need for choosing a suitable model that naturally offers reasonable computing efficiency with high detection accuracy. Drawing upon YOLO, we are seeking to help improve the efficacy of vehicle and free space detection systems to allocate more parking resources and enhance user experience in urban settings.

# Chapter 4

# Implementation And Results

In this chapter, we discussed about the implementation of various models we selected, their modification, which is applied to improve their reliability, their result and our analysis on them.

## 4.1 Model Implementation

The following describes the procedure with which we implemented the Faster RCNN, MobileNet SSD and YOLO models with python programming language. All the codes were run for both of the datasets separately. For the second dataset, we just modified the code a little, setting:

- `epoch` was set to 10 for each of the models.
- `learning rate` was set to 0.01 for each of the models.
- `batch size` was set to 16 for each.

Apart from those, it used the same code as the code which used the first dataset. The detailed description of each of the procedure is described below which was used to train the first dataset.

### 4.1.1 Faster RCNN

Specifically, in this study, the object detection model to detect cars and empty parking slots in image is constructed and tested. The model used is faster R-CNN with the ResNet-50 backbone and Feature Pyramid Network (FPN). These include data preprocessing, model building and model assessment, with more details described below:

**Data Preparation and Annotation Handling**

We store our dataset in a format that is specific to this task and follows the COCO data format which consists of images with their annotations. The dataset contains images with labeled bounding boxes for two classes of interest: Regardless, based on our observations of CAR and EMPTYSLOT, we proceed to the next set of experiments. The annotations also contain the IDs corresponding to the category of each object instance.

To facilitate data loading and preprocessing, we define a custom dataset class, `ParkingDataset`, which inherits from `torch.utils.data.Dataset`. This class performs the following operations:

- **Initialization**: Reads the necessary annotations in the COCO format and makes a list of image IDs.
- **Item Retrieval**: For every image in the class, the class pulls the actual image file and its accompanying annotations. It scans the image based on the PIL and transforms the image to an RGB type image.
- **Bounding Box Processing**: Returns fields, which contain bounding boxes from annotations. The bounding boxes are defined by the coordinates $(x_{\min}, y_{\min}, x_{\max}, y_{\max})$, calculated from the COCO bounding box format $(x, y, w, h)$.
- **Label Adjustment**: Maps the original category IDs to new labels: label 1 for *Car* (original IDs 2 or 3) and label 2 for *Empty Slot* (original IDs 1 or 4).
- **Tensor Conversion**: Transforms coordinates and annotations from bounding boxes and labels to PyTorch tensor and calculates other targets including area and crowd.
- **Transformations**: Pre-processes data, conducts conversion of images into tensors and flipping the images during training.

**Data Augmentation**
To enhance the model's generalization, we implement a transformation function, `get_transform`, which performs random horizontal flips with a 50% probability during training. When an image is flipped, the corresponding bounding boxes are adjusted accordingly by updating the $x$-coordinates using:

$$x'_{\min} = \text{width} - x_{\max}, \quad x'_{\max} = \text{width} - x_{\min} \tag{4.1}$$

**Model Architecture and Modification**
We specifically use Faster R-CNN model with a ResNet-50 as its backbone, and FPN as our default network since are pre-trained on the COCO dataset. As our task requires distinguishing between two objects and empty slots, in addition to the background, we modify the classifier of the model to output three classes. Specifically, we replace the pre-trained model's ROI heads' box predictor with a new predictor that has an output dimension corresponding to our number of classes:

$$\text{num\_classes} = 3$$

This adjustment is made by extracting the number of input features from the existing predictor and initializing a new `FastRCNNPredictor`:

$$\text{in\_features} = \text{model.roi\_heads.box\_predictor.cls\_score.in\_features}$$

**Training Procedure**
The model is trained using Stochastic Gradient Descent (SGD) with the following hyperparameters:

- Learning rate: $\eta = 0.001$
- Momentum: 0.9
- Weight decay: 0.0001
- Gradient clipping: Pass 1.0 to the maximum norm so that we don't have exploding gradients.

At each epoch the model takes batches of images along with the corresponding targets. The loss is computed as the sum of various loss components provided by the Faster R-CNN model, such as classification loss and bounding box regression loss:

$$\text{Total Loss} = \sum_i \text{Loss}_i \tag{4.2}$$

This is why to update the model parameters the optimizer minimizes the total loss of the model. It is updated after 50 iterations, and a record of how the progress is going is kept.

**Evaluation and Metrics Calculation**

For evaluation, we employ a separate test set, comprised of image and annotations in the same format. The evaluation process involves the following steps:

- **Model Loading**: The trained model trained is loaded and the state dict of the loaded model is set to an evaluation mode.
- **Image Processing**: Every test image is pre-loaded and normalized into a tensor form that the model requires for its examination.
- **Prediction**: Both the reference and the degrade model produce bounding boxes, labels, and confidence scores using the model's prediction.
- **Thresholding**: Predicted bounding boxes are therefore further filtered through a confidence score which may be set to 0.5 and above.
- **Intersection over Union (IoU)**: Finally, for each of the predicted bounding box, a measure of overlap with the ground bound boxes is computed using IoU metric. The IoU is calculated as:

  $$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} \tag{4.3}$$

  where coordinates of the predicted and actual 'boxes' can be cross multiply to obtain the area of intersection and union.
- **Matching Detections**: The post-analysis of the prediction is defined as a true positive predicted box, if its IoU with a GT box is greater than a fixed threshold 0.5 and has the same label as the GT box.

- **Metrics Computation**: In terms of accuracy, precision, recall, F1-score on one hand, we have the following formulas for every class:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{4.4}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{4.5}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.6}$$

**Visualization**
Bounding boxes of detected objects are superimpose on the image for purposes of validation. Predicted bounding boxes for *Car* class are depicted as green, while those of *Empty Slot* class are depicted as red color. The confidence score of the model is represented by the number displayed at the right of every box to show the level of confidence it has in its input.

### 4.1.2 MobileNet SSD

Here in this study, we design and assess an object detection model specifically working on detection of cars and non-occupied parking spaces within images. The model has been developed from the Single Shot MultiBox Detector (SSD) architecture with MobileNetV3-Large backbone, with implementation coming from the Pytorch `torchvision` package.
For processing the data which are divided according to the COCO format, we create a specific dataset class which reads the annotations and transforms the data into a form suitable for training and testing. Every single image within the developed dataset has attributes linked to annotations together with the bounding boxes and the category IDs. We focus on two object classes: The two symbols are then identified as *Car* that is labeled 1 and *Empty Slot* labeled 2. The background class is managed naturally and does not require going deeper into the scheme.
The bounding boxes in COCO format appearing in Table 2 are specified as $(x_{\min}, y_{\min}, w, h)$. We convert them to the format required by the model, which is $(x_{\min}, y_{\min}, x_{\max}, y_{\max})$, by computing:

$$x_{\max} = x_{\min} + \text{width}, \quad y_{\max} = y_{\min} + \text{height} \tag{4.7}$$

In data preprocessing we use a function that ensures that the images fed into the neural network are in the form of tensors. This is with the aid of the `transforms.ToTensor()` technique from the transforms package of the torchvision module.
The SSD model is initialized with the MobileNetV3-Large backbone and modified to accommodate three classes (background, car, empty slot) by specifying the number of classes during instantiation:

$$\text{model} = \text{ssdlite320\_mobilenet\_v3\_large}(\text{num\_classes} = 3)$$

The model is transfer to GPU for training in order to take full advantage of the hardware assistances.

The training process spans 67 epochs, where each epoch consists of the following steps:

1. **Data Loading**: A batch of images and corresponding targets (annotations) is loaded from the training dataset.

2. **Forward Pass**: The images are passed through the model to obtain predictions, including class probabilities and bounding box coordinates.

3. **Loss Computation**: The model calculates the loss, which comprises the classification loss (e.g., cross-entropy loss) and the localization loss (e.g., Smooth L1 loss). The total loss is given by:

$$\text{Total Loss} = \text{Classification Loss} + \text{Localization Loss} \tag{4.8}$$

Finally, as a part of the training progress display, we print the total loss after each epoch of training.
Finally for the purpose of evaluation, we employ a third test dataset and switch the model to its evaluation mode. We measure the quality of the model in terms of Intersection over Union (IoU) which calculates the overlapping of the Bounding boxes that has been predicted by the model and the actual Bounding Box. The IoU is calculated as:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{4.9}$$

where measurement of the overlap is the measure of the region of overlap of the predicted and ground truth bounding boxes, and the area of union is the area of both boxes combined minus the area of overlap.
We set the IoU threshold equal to 0.5 to classify a predicted bounding box as true positive image. For each class (*Car* and *Empty Slot*), we compute the following metrics:

- **True Positives (TP)**: The number of predicted boxes that localize the same object class as a ground truth box with intersection over union greater than the threshold.

- **False Positives (FP)**: Overlap between the number of predicted boxes and all the ground truth boxes of the same class for which the IoU is above the threshold.

- **False Negatives (FN)**: The number of ground truth image bounding boxes that have no corresponding predicted image bounding box with an intersection over union higher than the defined IoU threshold.

Using these values, we calculate precision, recall, and accuracy for each class:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{4.10}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{4.11}$$

$$\text{Accuracy} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}. \tag{4.12}$$

These metrics prove quite helpful when assessing the effectiveness of automatically identifying and categorizing objects by the model. A high precision means there are few false positives and a high recall means there are few false negatives.
As we described earlier to visualize the model 's predictions bounding boxes drawn on the test images. Segments corresponding to the location of the *Car* class are shown in green bounding boxes and the segments corresponding to the *Empty Slot* class are in red bounding boxes. The images are then written to an output directory for qualitative analysis.

### 4.1.3 YOLO

In this paper, we deploy and assess an object detection model, utilising YOLOv8 architecture, for detecting cars and unoccupied parking spaces in images. Data preprocessing and acquisition; model development; model assessment, and; result presentation and interpretation, as described below.
**Dataset Preparation**
First, let us organize the perceptions dataset, which we have in the COCO format to start with. The data set contains images and their annotation, and each annotation is comprised of the image's bounding boxes and category IDs of objects in the image. Our goal is to transform these annotations into a form that can be used to train this YOLOv8 model.
The steps for dataset preparation are:
1. **Loading COCO Annotations**: Using the annotations in the COCO formed JSON file that we have created, we load it using the COCO API. Every annotation has info about the image and the objects into it such as the four coordinates of the bounding box and the category that it belongs.
2. **Converting Annotations to YOLO Format**: Annotations in YOLO format must have a specific structure: each line represents one object and contains the class label and the ratio of the object's bounds to the image's size in the format:

$$\text{label} \quad x_{\text{center}} \quad y_{\text{center}} \quad \text{width} \quad \text{height}$$

where:
- label is the class index (e.g., 0 for empty slot, 1 for car). - $x_{\text{center}}$ and $y_{\text{center}}$ are the normalized center coordinates of the bounding box. - width and height are the normalized width and height of the bounding box.
The conversion from COCO format $(x, y, w, h)$ to YOLO format is done using:

$$\begin{aligned} x_{\text{center}} &= \frac{x + \frac{w}{2}}{\text{image\_width}}, \quad y_{\text{center}} = \frac{y + \frac{h}{2}}{\text{image\_height}} \\ \text{width} &= \frac{w}{\text{image\_width}}, \quad \text{height} = \frac{h}{\text{image\_height}} \end{aligned} \tag{4.13}$$

Then we make the conversion for each annotation and translate the previously mentioned category IDs to the corresponding class labels that we assigned.

3. **Creating YOLO Dataset Files**: After obtaining and pre-preprocessing the dataset which contains various images of the objects of interest, we arrange it in the way YOLO requires.
- All the images are placed in an `images` sub-directory. - A `labels` directory in which you have to put the text files with annotation data for every image.
Every image file in the folder `images` there is the folder `labels` with the base name of the file being the same.

**Model Training**

In this paper, we utilize the YOLOv8 model that is fast in operation while being highly accurate in its identification of objects. The training process involves the following steps:
1. **Model Initialization**: We start with a YOLOv8 model whose configuration file is `yolov8n.yaml` that specifies this architecture and hyperparameters for a small and fast version of YOLOv8.
2. **Training Configuration**: We define a YAML configuration file where we set the paths to the two sets, the numer of classes and the names of the classes:

$$\begin{aligned} \text{path}: &\quad [\text{dataset\_root\_directory}] \\ \text{train}: &\quad [\text{relative\_path\_to\_train\_images}] \\ \text{val}: &\quad [\text{relative\_path\_to\_validation\_images}] \\ \text{nc}: &\quad 2 \quad (\text{number of classes}) \\ \text{names}: &\quad [\text{'empty\_slot'}, \text{'car'}] \end{aligned}$$

3. **Model Training**: We fit the model for 100 epochs on the prepared data and with the configuration set above. The training process reduces the detection and classification loss of the model in the training dataset by adjusting model weight.

**Model Evaluation**

In this work, we assess the accuracy of the trained model on the test set acquired from the web. Another criterion is the precision (precision), return rate or recall (recall), as well as the mean Average Precision (mAP) characteristic, which is used in the evaluation of object detection.
1. **Running Validation**: For evaluation, we make use of the validation method provided in the model to predict on the test set and compute metrics. The expected outputs of the model are bounding boxes for the images, their corresponding classes and confidence levels.
2. **Calculating Metrics**:
- **Precision**: Calculates the ratio of the number of correct positive observations, to the total number of positive observations which have been predicted.

$$\text{Precision} = \frac{\text{True Positives}}{(\text{True Positives} + \text{False Positives})} \tag{4.14}$$

- **Recall**: -One measures the number of correctly predicted positive observations to the total number of positive observations in the sample.

$$\text{Recall} = \frac{\text{True Positives}}{(\text{True Positives} + \text{False Negatives})} \tag{4.15}$$

- **Mean Average Precision (mAP@50)**: By using the average precision, and taking the mean of those scores, calculated from the model at an IoU of 0.5. It is widely used up-to-date for assessment of object detection models.

3. **Confusion Matrix**: For class-level evaluation, we extract the confusion matrix to determine the effectiveness of the model from each class view point. Confusion matrix helps to understand the true positive, false positive and false negatives for each of the class.

**Result Visualization**

In order to evaluate the model qualitatively we draw bounding boxes on the test images to illustrate the results of the model described above.

1. **Drawing Bounding Boxes**: In this case, for every detected bounding box, we obtain the co-ordinates, class label, and the confidence estimate. We draw rectangles on the images corresponding to the bounding boxes:

- **Color Coding**: To do this, we need to assign different colours to different classes as for instance green for empty slot while red for cars.

2. **Annotating Images**: Furthermore we annotate each of the bounding boxes with the class label and the confidence score achieved by the model for improved visualization.

3. **Saving Output Images**: The images are then annotated and information about them stored to an output directory for analysis and presentation.

## 4.2 Applying Modification

This section discussed about how we improved an existing model by applying some modification in the running process. After running all three models, we tried to improve the free space detection performance of YOLOv8 model. Note that this modification can be applied to any other model as well. We used the same parameters of the codes of respective datasets to apply modification for each of the datasets separately. To do this, we aimed to insert more information in the images to ensure that each bounding box of empty slot also contains some information about the distance between closest car and the scale of the images. We followed the following procedure in order to achieve this.

1. **Image Modification**

- First, from the training images, we saved the midpoint position of bounding boxes of cars in a list, and inserted the data into a dictionary in `img_name :list_of_midpoints` format.

- We defined a variable named `scale` and a function `color_func`, which takes a paramerter named `dist`, which is a floating number. It returns the value of

$$\text{color\_func} = 255 \cdot \sin^{\frac{1}{3}}\left(\frac{\text{dist} \cdot \pi}{2000}\right) \tag{4.16}$$

- Then we loaded the image in with `PIL Image` function, then converted it into a multi dimensional array. For each non negative integer $(i, j)$, we select $(\text{scale} \cdot i, \text{scale} \cdot j)$ positioned pixel if it is inside the image. Then we calculate the minimum euclidean distance from this position to all the midpoints of cars in the image. We save it to `dist` and pass it to `color_func` function, and color the pixel with $\text{RGB}(0, \text{color\_func}(\text{dist}), 0)$ color. If the $2 \times 2$ box with this pixel at the top left is inside the image, we color the whole $2 \times 2$ box with

the same color. Then we convert back the multi dimensional array to image and save it to a different folder to run training function later.

- Then we put all the path of testing images in a list and iterated it to pass the path in the model. Then we got output of all the bounding boxes. We separated the bounding boxes of cars from the rest and color it with the same procedure as we did for the training images. Then we save all the modified images in a different folder to run test for later.

2. **Re-run the Model**
Then we use the previously defined functions to re-run the whole YOLO code. But in this case we use the modified image folders instead of the previous images.
3. **Change `scale` and Evaluate**
After comparing result with previous one, we change the value of `scale` and run the second part of the code including the image modification again. Do it for different values of `scale`.

Figure 4.1: Combining Car positioning image with the original with `scale=20` to get the modified image.

## 4.3 Performance Evaluation

We analyzed the performance of all the models and modification we ran so far in this section.

### 4.3.1 Precision and Recall Metrics

The performance of the three object detection models—Faster R-CNN, MobileNet SSD, and YOLO—was evaluated based on their precision and recall in detecting

cars and empty parking slots. The results are summarized in Tables 4.1 and 4.2 for the first dataset, and Tables 4.3 and 4.4 for the second dataset.

| **Model** | **Precision (%)** | **Recall (%)** |
|---|---|---|
| Faster R-CNN | 96.8 | 88.97 |
| MobileNet SSD | 58.04 | 80.43 |
| YOLO | 96.1 | 97.5 |

Table 4.1: Precision and Recall for Car Bounding Boxes of the First Dataset

| **Model** | **Precision (%)** | **Recall (%)** |
|---|---|---|
| Faster R-CNN | 84.96 | 100 |
| MobileNet SSD | 43.58 | 61.39 |
| YOLO | 90.2 | 82.6 |

Table 4.2: Precision and Recall for Empty Slot Bounding Boxes of the First Dataset

| **Model** | **Precision (%)** | **Recall (%)** |
|---|---|---|
| Faster R-CNN | 96.79 | 86.47 |
| MobileNet SSD | 46.62 | 68.61 |
| YOLO | 94.5 | 99.3 |

Table 4.3: Precision and Recall for Car Bounding Boxes of the Second Dataset

| **Model** | **Precision (%)** | **Recall (%)** |
|---|---|---|
| Faster R-CNN | 95.03 | 91.38 |
| MobileNet SSD | 40.94 | 63.56 |
| YOLO | 98.9 | 94.3 |

Table 4.4: Precision and Recall for Empty Slot Bounding Boxes of the Second Dataset

### 4.3.2 Visual Comparison of Precision and Recall

To provide a visual comparison of the precision and recall metrics across different models for both car and empty slot detection, Figure 4.2 presents a bar diagram illustrating these performance indicators for the first dataset. The figure 4.3 represents the second dataset. The common parameters of the models of the second dataset is the same. So it provides a comparatively accurate comparison among them.

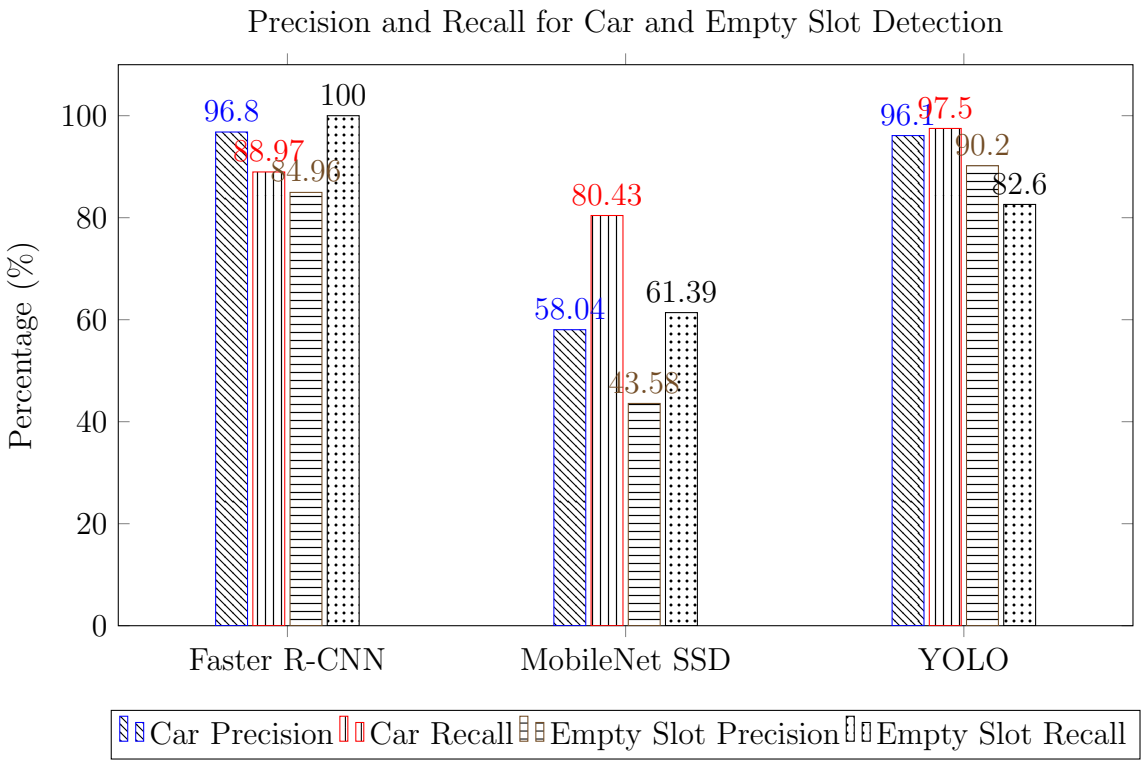



Figure 4.2: Comparison of Precision and Recall for Car and Empty Slot Detection across Different Models of the First Dataset

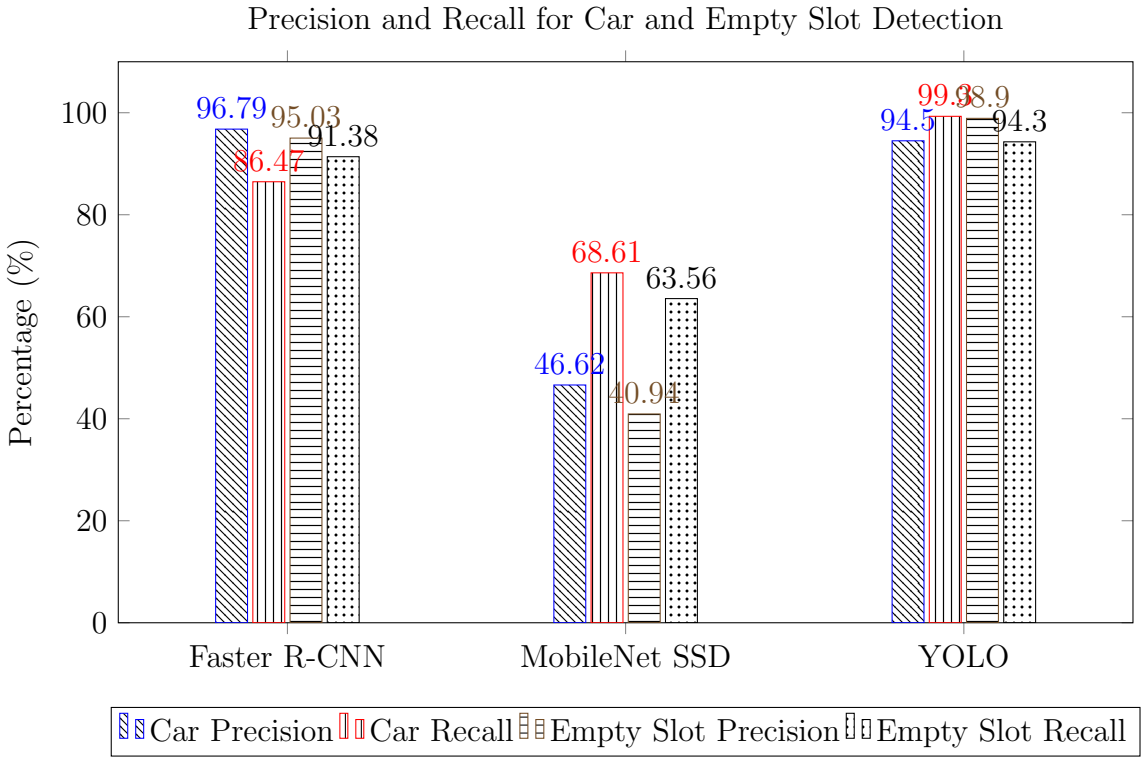



Figure 4.3: Comparison of Precision and Recall for Car and Empty Slot Detection across Different Models of the Second Dataset

### 4.3.3 Enhanced YOLO Model with Different Scale Modifications

To further improve the YOLO model's performance, several modifications were made based on variable named `scale`. The precision and recall metrics for car and empty slot bounding boxes of the first dataset across different scale values are presented in Tables 4.7 and 4.8, respectively. The 4.7 and 4.8 tables presented the values for the second dataset.

Table 4.5: Precision and Recall for Car Bounding Boxes of First Dataset with Different Scale Values

| **Scale** | **Precision (%)** | **Recall (%)** |
|---|---|---|
| Base | 96.1 | 88.7 |
| Scale=4 | 96.0 | 97.9 |
| Scale=8 | 96.3 | 98.8 |
| Scale=16 | 96.9 | 97.6 |
| Scale=20 | 97.8 | 97.7 |
| Scale=24 | 96.6 | 98.1 |
| Scale=32 | 95.5 | 97.8 |

Table 4.6: Precision and Recall for Empty Slot Bounding Boxes of First Dataset with Different Scale Values

| **Scale** | **Precision (%)** | **Recall (%)** |
|---|---|---|
| Base | 90.2 | 82.6 |
| Scale=4 | 89.7 | 80.1 |
| Scale=8 | 88.4 | 83.3 |
| Scale=16 | 91.3 | 84.2 |
| Scale=20 | 92.1 | 81.9 |
| Scale=24 | 90.7 | 83.2 |
| Scale=32 | 88.5 | 83.6 |

Table 4.7: Precision and Recall for Car Bounding Boxes of Second Dataset with Different Scale Values

| **Scale** | **Precision (%)** | **Recall (%)** |
|---|---|---|
| Base | 94.5 | 99.3 |
| Scale=8 | 93.8 | 99.5 |
| Scale=16 | 94.6 | 99.3 |
| Scale=20 | 94.3 | 99.3 |
| Scale=32 | 94.6 | 99.3 |

Table 4.8: Precision and Recall for Empty Slot Bounding Boxes of Second Dataset with Different Scale Values

| **Scale** | **Precision (%)** | **Recall (%)** |
|---|---|---|
| Base | 98.9 | 94.3 |
| Scale=8 | 98.9 | 93.3 |
| Scale=16 | 98.8 | 94.2 |
| Scale=20 | 98.7 | 94.1 |
| Scale=32 | 98.8 | 94.3 |

### 4.3.4 Visual Comparison of Precision and Recall of Scale Modifications

The following diagram shows the comparison between the base and different versions of YOLO diagram for each of the dataset.

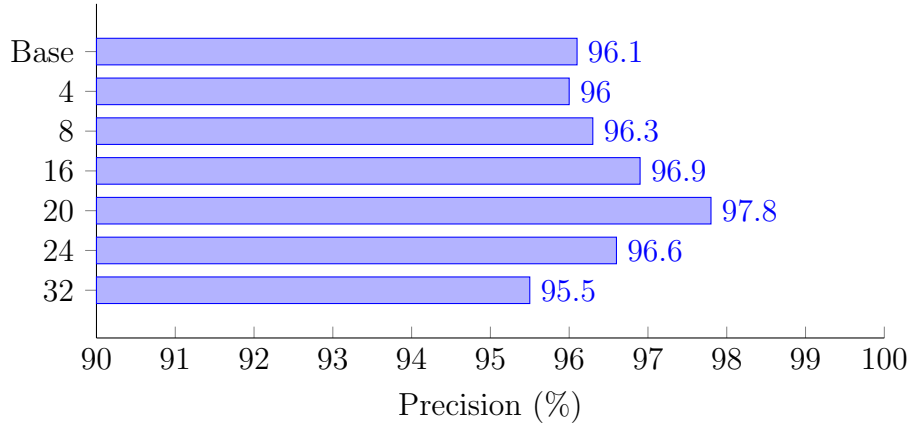



Figure 4.4: Car bounding box precision of first dataset at different scales

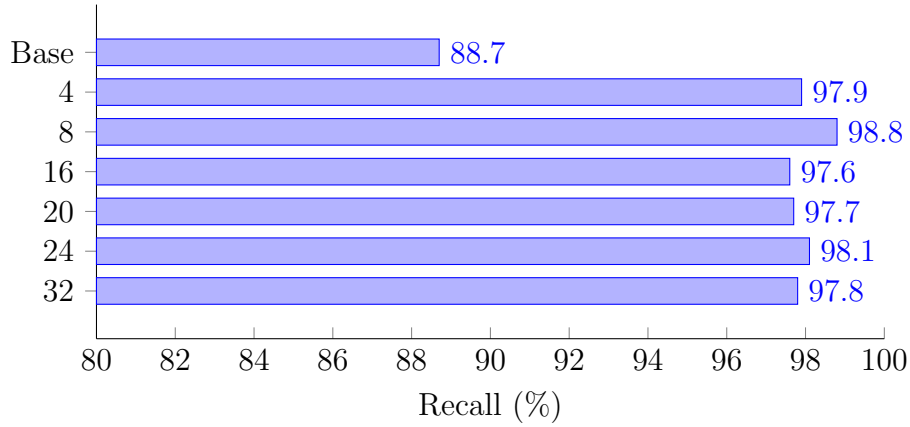



Figure 4.5: Car bounding box recall of first dataset at different scales

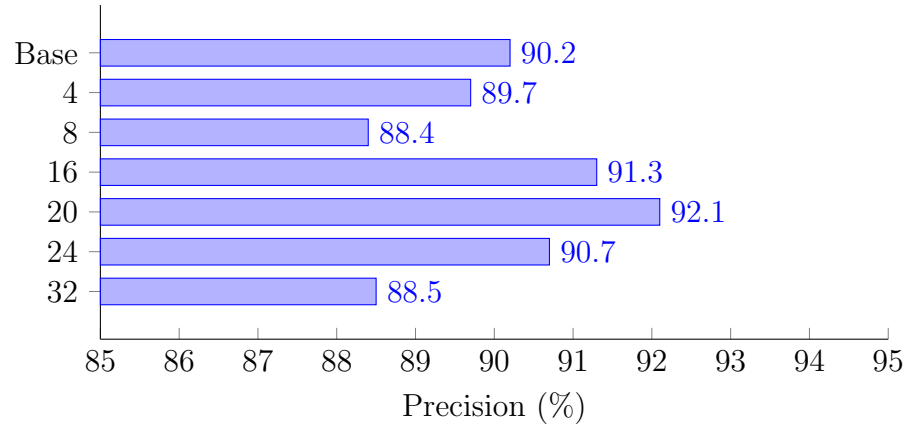



Figure 4.6: Empty slot bounding box precision of first dataset at different scales

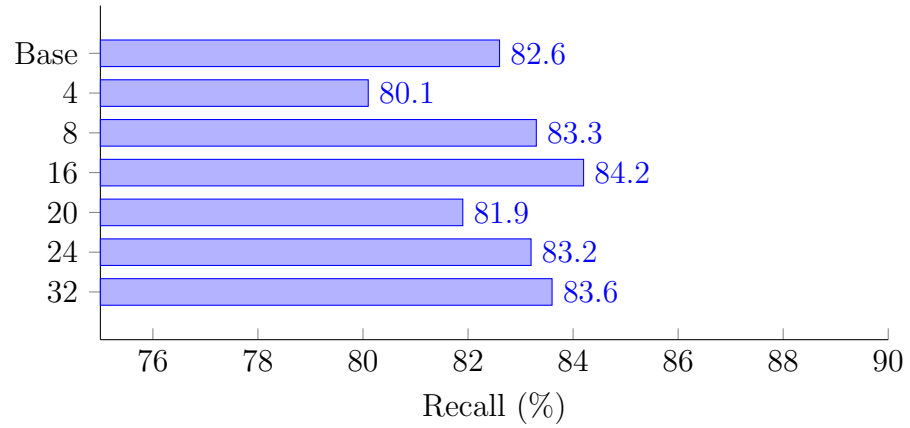



Figure 4.7: Empty slot bounding box recall of first dataset at different scales

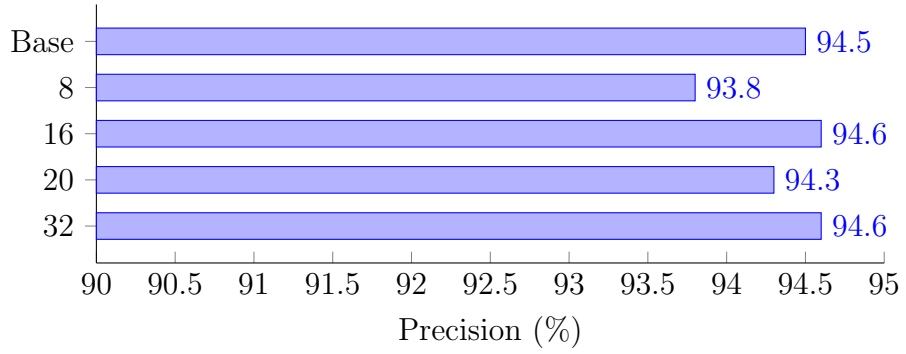



Figure 4.8: Car bounding box precision of second dataset at different scales

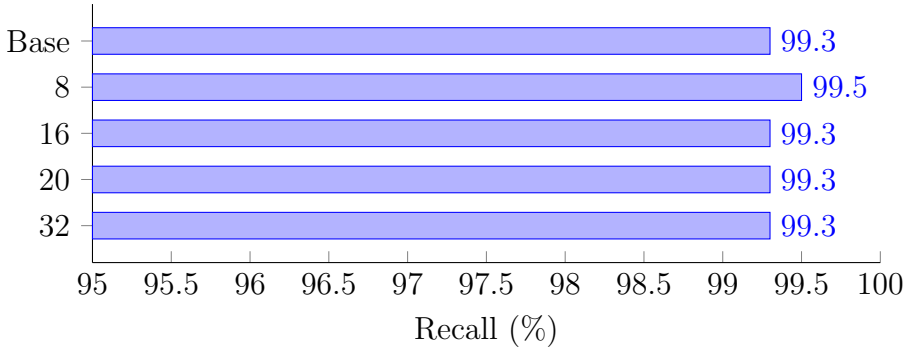



Figure 4.9: Car bounding box recall of second dataset at different scales

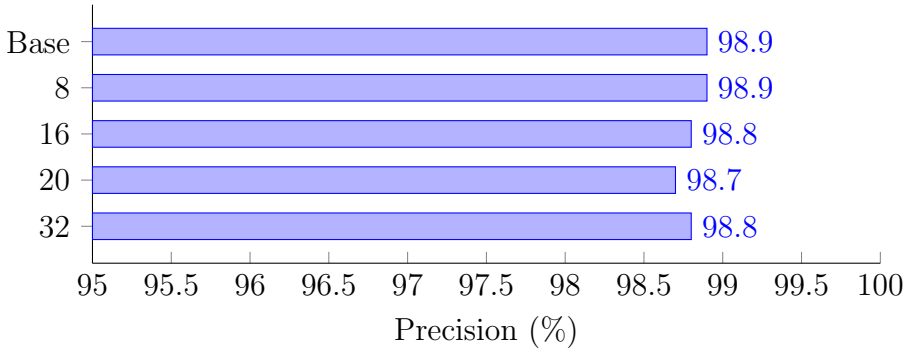



Figure 4.10: Empty slot bounding box precision of second dataset at different scales

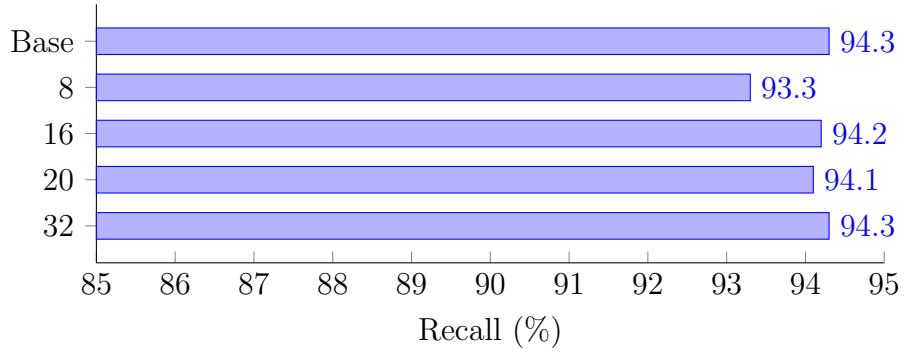



Figure 4.11: Empty slot bounding box recall of second dataset at different scales

## 4.4 Analysis of Overall Results

From the above data, we can find the following results for the first dataset. Since the specialty of first dataset is it's smaller size, this is applicable for any smaller size dataset, where the garage owner wants the customized trained model for their garage, which significantly helps to set up a more accurate prediction model for the specific garage with lower effort.

- For car bounding box's, precision, Faster RCNN has the highest precision, which is 96.8%. YOLO is right behind with 96.1%. But, for recall, YOLO is in the first place with 97.5% where Faster RCNN is only 88.97%.
- For empty slot's bounding box precision, YOLO has the highest value, 90.2%, where second Faster RCNN has 82.6%. But for recoll, Faster RCNN has complete 100%, where YOLO has 84.96%. So we can see that even though Faster RCNN has 100% success rate in detecting correct slot, gives a lot of false positive information about car bounding box. On the other hand, YOLO is balanced.
- Because YOLO gives a stable result and also comparatively fast, it was chosen for improvement. **After modification, scale=20 gives highest car precision, which is 97.8% and empty box precision, which is 92.1%. Which has clearly 1% improved from previous best Faster RCNN car precision of 96.8%, and 1.9% improvement from YOLO empty slot precision of 90.2%. So, we can say if this modification declares a bounding box to be car or empty box, it has the highest chance that the statement is true. So, it is a clear improvement from all the previous model we experimented so far. So, our medication with scale=20 is our final version.**
- Aside from `scale` value 20, 16 gives a more stable result. It improves precision and recoll of both cars and empty slots, where `scale=20` gives lower rcall for empty slots bounding boxes.
- The modification keeps the computational complexity as it is. But it increases the execution time since the model is running twice.
- MobileNet SSD gave a very poor result for all the section. We tried to improve it by changing various parameters, but result did not vary remarkably.

Below contains the summarized result for the second dataset. As it's size is comparatively large, custom trained model for a specific garage is not available here. So it is for more generalized senerio, where garage owner wants to use the model provided as they are. So the prediction result may significantly vary based on garage structure. But from our dataset, the analysis on it is specified below.

- For car bounding box, Faster R-CNN has the highest precision, 96.79%, where YOLO is close, 94.5%. But for Recall, YOLO is significantly ahead of Faster R-CNN, which is 99.3%, way ahead of Faster R-CNN, 86.47%.
- For empty box, Faster R-CNN has precision 95.03%, second to YOLO, which is 98.9%. For recall, again, YOLO is ahead of Faster R-CNN's 91.38%, having 94.3% prediction recall.

- MobleNet SSD has the least precision and recall for both cars and empty slot prediction. Comparing between Faster R-CNN and YOLO, YOLO's overall performance is higher than Faster R-CNN by a clear margin.

- Unlike the first dataset, the influence of integrating spatial position of cars in the image in different scale has a very small influence on the second dataset. The reason is the variety of size of cars, empty slots and environment can't influence the judgment of empty slot much as the data is way diverse. But, some targeted improvement is achievable, for example, `scale=32` increased car precision by 0.1%, sacrificing empty box's precision by the same amount. But except `scale=8`, the maximum change in percentage did not exceed 0.2%, which is negligible.

# Chapter 5

# Conclusion

## 5.1 Deployment Procedure

By testing the first datset, to deploy this improved model in a garage, only a couple of CCTV images of a garage is needed. This can be easily collected from the same CCIV footage of the garage. After training, the testing function can easily analyze the car bounding boxes and empty slots for 20 to 30 fps videos depending on computing power of the devices. The main advantage of this research is the modification is not limited to YOLO model. It can be applied to any model. But finding suitable scale for each of those models may require more effort. Even though the modification does not change the initial computational complexity, this modification trades off execution time for performance. So this is also something to consider. But for CCTV footage analysis, it should not be a big concern since limiting fps for a video will not change the parking management system much.

## 5.2 Limitation and Future Work

If technology and time approved, we could use more cutting edge models to use in this project. But the lack of documentation and forum posts of recent models were main hindrance. Also, our devices were not suitable to run heavy machine learning codes, so we had to use shared PC from our university. So we could get a little time to debug the codes as the time slot was only 16 hours per week. We could also change the `color_func` function and try to detect a better function which works better. Because of the same limitation, it was not possible. We also had to predict where the local maximum precison can be found to detect the `scale=20` value. So, in the future, this project can be enhanced the following ways:

- Using more cutting edge technology to select the model to enhance. Since our enhancement is not dependent on any specific model, we just need to experiment with the optimal scale to find the correct value we need.
- Experimenting with more color function for modification of the images.
- Moderately modifying the model so that it can detect cars and empty slots in real time.

- Instead of using minimum euclidean distance to define the function, different kind of distance could be used. Since car is not round, but square shpaed, they may have a higher possibility of giving better results.

There is infinite possibility in this modification idea. Only by spending more time and experimenting with it, more accurate result could be found.

## 5.3 Conclusion

The access to car ownership in today's world scenario has been tremendously raising the bar for correct spotting of appropriate parking areas more and more difficult without utilization of automated management systems. At the same time, more extensive construction of buildings, industries, and institutions around the world, which contributed to the increased congestion of automobile traffic and reduced coverage of parking areas, increased the divergence between the increasing number of vehicles and the availability of suitable parking space. In view of these advancements, therefore, it is the spirit of this research to enhance the creation of an effective and proficient machine learning model to minimize the operating expenses of the garage owners and help the car owners in finding the available space for parking. In this study, we use the Faster R-CNN model to detect the parked cars and the open parking spaces in garages. In addition to helping vehicle owners enjoy a smooth drive to the parking lot, and through the use of enhanced object detection methodologies, this paper assists both commercial and non-commercial garage management to automate operational functions. The application of information technology in parking management systems results in improvements in the operational and labor costs for the garages, thus lifting the profitability of its operations. The suggested solution makes managing the garage far more efficient and less complicated to handle in terms of execution by the clients. This increase in efficiency makes private garage owners access their stations to the public, thus enhancing the availability of parking lots. Moreover, thanks to low entry barriers, this system encourages a range of companies to introduce automated parking services, which potentially can be supported and realized by governmental institutions to improve the quality of urban parking systems. Effective parking space management in fact, will not only enhance management of the available resources but will also assist in reducing traffic explosions as well as pollution in the cities. Furthermore, this research gives a strong background to build a model for research that deals with other issues in the management of parking. This generalizability and modularity of the methodology can be highlighted in several ways that makes it convenient for other researchers to extend from our study and apply it in different contexts and settings. This research proposes a viable and affordable solution that, as the scale is extended, enables further integration of intelligent parking management systems into urban transport systems, thus enabling the creation of intelligent transportation systems. Consulting the results of this research, it can be concluded that machine learning, in particular Faster R-CNN, can become a tool that will significantly change parking management systems. In addition to helping garages work faster and more efficiently, our model also increases the satisfaction of vehicle owners by improving the accuracy of the vehicle and free slot detection. This ability to advance technological solutions and create societal panacea that also provide dual benefits is indeed one of the ways

through which technologists have ensured that advanced technological solutions are embedded within the fabric of everyday infrastructure leading to intelligent urban living architecture.

# Bibliography


[1] ResearchGate, *Effects of car involvement in congestion and road accident in dhaka city: A study on sustainable solutions*, https://www.researchgate.net/figure/Composition-of-registered-vehicles-in-Dhaka-city_fig2_277313648 [Accessed: 21 Jan, 2024], 2012.

[2] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, *Selective search for object recognition*, Journal Article, Accessed on 2024-10-17, Apr. 2013.

[3] P. R. De Almeida, L. S. Oliveira, A. S. Britto Jr, E. J. Silva Jr, and A. L. Koerich, "Pklot–a robust dataset for parking lot classification," *Expert Systems with Applications*, vol. 42, no. 11, pp. 4937–4949, 2015.

[4] R. Girshick, *Fast r-cnn*, https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf, Conference Paper, Accessed on 2024-10-17, Dec. 2015. [Online]. Available: https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf.

[5] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, https://arxiv.org/pdf/1506.01497.pdf, Conference Paper, Accessed on 2024-10-17, Dec. 2015. [Online]. Available: https://arxiv.org/pdf/1506.01497.pdf.

[6] W. Liu, D. Anguelov, D. Erhan, *et al.*, *Ssd: Single shot multibox detector*, https://arxiv.org/abs/1512.02325, Conference Paper, Accessed on 2024-10-17, Oct. 2016. [Online]. Available: https://arxiv.org/abs/1512.02325.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, https://arxiv.org/abs/1506.02640, Conference Paper, Accessed on 2024-10-17, Jun. 2016. [Online]. Available: https://arxiv.org/abs/1506.02640.

[8] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, and T. Weyand, *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, https://arxiv.org/abs/1704.04861, Preprint, Accessed on 2024-10-17, Apr. 2017. [Online]. Available: https://arxiv.org/abs/1704.04861.

[9] S. Addala, *Research paper on vehicle detection and recognition*, May 2020. DOI: 10.13140/RG.2.2.34908.82561.

[10] Y. Liu, X. Li, and X. Liang, *A survey on object detection in parking lot environments*, Journal Article, Accessed on 2024-10-17, Apr. 2020.

[11] E. Shreyas, M. H. Sheth, *et al.*, "3d object detection and tracking methods using deep learning for computer vision applications," in *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, IEEE, 2021, pp. 735–738.

[12] M. Tondra and E. Karami, "Vehicle detection and tracking," Ph.D. dissertation, Jul. 2021. DOI: 10.13140/RG.2.2.15030.22085.

[13] J. Almeida, S. Guamán, and S. G. Yoo, "Vechicle counting system in urban areas: A practical case," in *2022 IEEE 7th International conference for Convergence in Technology (I2CT)*, IEEE, 2022, pp. 1–6.

[14] T. A. Nayeem, S. Motaharuzzaman, A. T. Hoque, and M. H. Rahman, "Computer vision based object detection and recognition system for image searching," in *2022 12th International Conference on Electrical and Computer Engineering (ICECE)*, IEEE, 2022, pp. 148–151.

[15] S. Bilotta, L. A. I. Palesi, and P. Nesi, "Predicting free parking slots via deep learning in short-mid terms explaining temporal impact of features," *IEEE Access*, 2023.

[16] C. Data, *Bangladesh motor vehicle registered: Dhaka: Total*, https://www.ceicdata.com/en/bangladesh/motor-vehicle-registered/motor-vehicle-registered-dhaka-total [Accessed: 21 Jan, 2024], 2023.

[17] R. Kaur, R. K. Roul, and S. Batra, "A hybrid deep learning cnn-elm approach for parking space detection in smart cities," *Neural Computing and Applications*, vol. 35, no. 18, pp. 13 665–13 683, 2023.

[18] A. Khalfi and M. Guerroumi, "Transfer learning with image data augmentation for parking occupancy detection," in *2023 International Conference on Advances in Electronics, Control and Communication Systems (ICAECCS)*, IEEE, 2023, pp. 1–4.

[19] A. Martynova, M. Kuznetsov, V. Porvatov, *et al.*, "Revising deep learning methods in parking lot occupancy detection," *arXiv preprint arXiv:2306.04288*, 2023.

[20] D. Matiunina, N. Sautter, A. Loder, and K. Bogenberger, "Predicting parking occupancy with deep learning on noisy empirical data," in *2023 8th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, IEEE, 2023, pp. 1–6.

[21] Q. Cai, R. Yang, M. Wen, W. Huang, and J. Gu, "Painet: Towards fast and efficient parking lot lane detection," *IEEE Access*, 2024.

[22] T. A. Dheeven, P. M. Kumar, V. Venkatesh, and K. I. Sailaja, "Iot based sensor enabled vehicle parking system," *Measurement: Sensors*, vol. 31, p. 100 953, 2024, ISSN: 2665-9174. DOI: https://doi.org/10.1016/j.measen.2023.100953. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2665917423002891.

[23] F. Hasan Yusuf and M. A Mangoud, "Real-time car parking detection with deep learning in different lighting scenarios," *International Journal of Computing and Digital Systems*, vol. 15, no. 1, pp. 1–9, 2024.

[24] A. H. Hidayah, S. Zuhriyah, B. E. W. Asrul, Y. Yuyun, E. Prakasa, *et al.*, "Integration of yolov5 algorithm and opencv in innovative smart parking management approach," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 8, no. 3, pp. 413–422, 2024.

[25] N. Himabindu, S. Vishnu, M. Vishveswaran, R. Swaroop, and B. Bairwa, "Design and implementation of rfid-driven parking management systems," in *2024 IEEE International Conference on Big Data & Machine Learning (ICBDML)*, IEEE, 2024, pp. 97–102.

[26] D. Neupane, A. Bhattarai, S. Aryal, M. R. Bouadjenek, U. Seok, and J. Seok, "Shine: A deep learning-based accessible parking management system," *Expert Systems with Applications*, vol. 238, p. 122 205, 2024.

[27] B. N. Portal, https://www.brta.gov.bd/site/page/74b2a5c3-60cb-4d3c-a699-e2988fed84b2/Number-of-registered-Vehicles-in-Whole-BD [Accessed: 21 Jan, 2024], 2024.

[28] M. Project, *Parking slot detection dataset*, https://universe.roboflow.com/major-project-drvlm/parking-slot-detection-htuej, Open Source Dataset, visited on 2024-10-17, Mar. 2024. [Online]. Available: https://universe.roboflow.com/major-project-drvlm/parking-slot-detection-htuej.

[29] M. Safran, A. Alajmi, and S. Alfarhood, "Efficient multistage license plate detection and recognition using yolov8 and cnn for smart parking systems," *Journal of Sensors*, vol. 2024, no. 1, p. 4 917 097, 2024.

[30] P. Selvam, P. Saravanan, M. Marimuthu, M. Nithya, V. Sathiyapriya, and V. N. Harsha, "Psdnet: A breakthrough parking space detection network powered by yolov8," in *2024 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, IEEE, 2024, pp. 1–7.

[31] V. Shankar, V. Singh, V. Choudhary, S. Agrawal, and B. Awadhiya, "An enhanced car parking detection using yolov8 deep learning capabilities," in *2024 First International Conference on Electronics, Communication and Signal Processing (ICECSP)*, IEEE, 2024, pp. 1–5.

[32] J. Singh, G. M. C. Gupta, S. D. Rayyan, G. Madhusudhan, N. Pratik, and A. Singh, "Design and fabrication of iot based car parking," in *E3S Web of Conferences*, EDP Sciences, vol. 556, 2024, p. 01 016.

[33] R. Singh, S. Goyal, S. Agarwal, and S. Upadhyay, *Evaluating the performance of ensembled yolov8 variants in smart parking applications for vehicle detection and license plate recognition under varying lighting conditions*, 2024.

[34] D. Sivaganesan, C. Reshmah, S. Swathy, and S. Mugilan, "Parking slot identification system using svm classifier and hog features," in *2024 Second International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI)*, IEEE, 2024, pp. 794–800.

[35] S. Sulaiman, T. A. Nasarudin, A. A. Aziz, and A. H. A. Razak, "Vacant parking guidance for open parking area via image processing technique," in *AIP Conference Proceedings*, AIP Publishing, vol. 2898, 2024.

[36] N. Thakur, E. Bhattacharjee, R. Jain, B. Acharya, and Y.-C. Hu, "Deep learning-based parking occupancy detection framework using resnet and vgg-16," *Multimedia Tools and Applications*, vol. 83, no. 1, pp. 1941–1964, 2024.

[37] Z. Xu, X. Tang, C. Ma, and R. Zhang, "Research on parking space detection and prediction model based on cnn-lstm," *IEEE Access*, 2024.

[38] W. Yang, D. Li, W. Xu, and Z. Zhang, "A lidar-based parking slots detection system," *International Journal of Automotive Technology*, vol. 25, no. 2, pp. 331–338, 2024.

[39] N. Zhao, K. Wang, J. Yang, F. Luan, L. Yuan, and H. Zhang, "Cmca-yolo: A study on a real-time object detection model for parking lot surveillance imagery," *Electronics*, vol. 13, no. 8, p. 1557, 2024.