# PARKING LOT AVAILABILITY PREDICTION WITH YOLOv4

A project report submitted in partial fulfillment of the requirements for the award of the degree of

**B.Tech.**

**in**

**Electronics and Communication Engineering**

By

**KOTTAPU NIKITHA REDDY (620151)**

**KOMMIDI MANIKEERTHANA REDDY (620149)**

**CHERYALA HARIKA(620120)**

**EC449**
**MAJOR PROJECT-B**
**NATIONAL INSTITUTE OF TECHNOLOGY**
**ANDHRA PRADESH-534101**

**MAY 2024**

## BONAFIDE CERTIFICATE

This is to certify that the project titled **PROJECT NAME** is a bonafide record of the work done by

**KOTTAPU NIKITHA REDDY (620151)**

**KOMMIDI MANIKEERTHANA REDDY (620149)**

**CHERYALA HARIKA(620120)**

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **ECE** of the **NATIONAL INSTITUTE OF TECHNOLOGY, ANDHRA PRADESH**, during the year 2023-2024.

| | |
|---|---|
| **Dr.A.Arun Kumar** | **Dr.S.Yuvaraj** |
| Project Guide | Head of the Department |

# ABSTRACT

Traditional parking systems typically experience inefficiencies, demanding manual searches for vacant spaces and requiring significant labour. This paper describes a unique strategy for autonomously mapping parking places inside a lot using the YOLOv4 object detection algorithm. YOLO is a SOTA object detector that uses a clever convolutional neural network to identify, categorise, and localise objects in images using annotations. This study uses YOLOv4 to recognise numerous objects in images and videos for traffic monitoring applications, trained on a customised dataset. The primary goals are to reduce human labour while increasing parking management efficiency. Through extensive testing, our technique shows a steady improvement in locating parking spots over time. Notably, the system performs admirably in busy lots during peak hours. By tackling this key gap in computer vision-based artificial intelligence, our study helps to progress intelligent parking solutions, which have the potential to revolutionize urban transportation and improve user experience.This model aims to enhance detection accuracy while enabling real-time processing without compromising speed, offering a more economically viable solution for efficient parking space detection.

*Keywords*: You Only Look Once (YOLO), Artificial Intelligence(AI) Autonomous parking space mapping, Computer vision (CV), State-of-the-art (SOTA).

**ACKNOWLEDGEMENT**

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

**Dr. A. ArunKumar**, our project guide, for helping us and guiding us in the course of this project .

**Dr. S. Yuvaraj**, the Head of the Department, Department of ECE.

Our internal reviewers, **Ch. Chaitanya Krishna** , **Dr. A. Krishna Chaitanya** , **Mr. T. Sanath Kumar** for their insight and advice provided during the review sessions.

We would also like to thank our individual parents and friends for their constant support.

# TABLE OF CONTENTS

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The rising number of vehicles has led to a corresponding increase in parking facilities. Many parking lots currently rely on ground sensors for space monitoring, employing traditional methods such as ultrasonic, geomagnetic, and infrared ray technologies. However, implementing and maintaining these sensors in each parking space, particularly in large lots, can be a costly endeavor. Despite delivering high accuracy, the associated expenses prompt the need for a more cost-effective alternative. To address this, our paper introduces an object detection model based on YOLOv4. This model aims to enhance detection accuracy while enabling real-time processing without compromising speed, offering a more economically viable solution for efficient parking space detection.

Several items in digital photos and movies may be identified and located using object detection technology. There are several well-established methods for object detection. While some of them are intended for greater speed, others are intended for greater precision [1]. This cutting-edge technology may be used to autonomous systems, parking support, item counting, and other applications. The usage of conventional graphics processing units (GPUs) for object detection is very beneficial and reasonably priced. The most accurate current neural networks require a high number of GPUs in order to train with a big mini-batch size.

YOLOv4, an advanced deep neural network, is among the fastest and most accurate techniques available for object detection from digital photos. To train in YOLOv4, one needs only a standard GPU [2]. With the help of the YOLOv4 algorithm, which was trained on a portion of the UFPR05 parking lot dataset, the thesis attempts to categorise and locate two items.

## 1.1 Problem Statement

### 1.1.1 Addressing Urban Parking Challenges

Urban areas worldwide are grappling with a persistent issue: the increasing imbalance between vehicle numbers and available parking spaces. This discrepancy manifests as a daily struggle for drivers, who often find themselves circling congested streets in search of elusive parking spots. Traditional parking management systems, reliant on manual checks or rudimentary sensors, fail to provide efficient and accurate solutions to this pressing problem.

Figure 1.1: A bustling urban street with vehicles navigating through traffic

### 1.1.2 Identifying Shortcomings in Current Approaches

Existing parking management systems exhibit several shortcomings:

- **Inefficiency:** Manual monitoring of parking spaces is prone to human error, while basic sensor technologies often struggle to discern between occupied and vacant spots, resulting in unreliable data.

- **Real-time Updates Deficiency:** Traditional systems lack the capability to deliver instantaneous updates on parking availability, leaving drivers unaware of recent changes in occupancy status.

- **Scalability Challenges:** Implementing these solutions in sprawling parking lots or multilevel structures proves complex and cost-prohibitive, limiting their widespread adoption.

These limitations underscore the critical need for a more robust, adaptable, and technologically advanced parking management system capable of delivering real-time insights and optimizing parking utilization.

## 1.2 Motivation

### 1.2.1 Elevating Efficiency and User Experience

The deployment of a car detection system using YOLOv4 offers a promising avenue to tackle the challenges of parking management head-on. This system stands to revolutionize parking operations by:

- **Automating Spot Detection:** Leveraging the advanced object detection capabilities of YOLOv4, the system can autonomously identify and locate vehicles within parking facilities, eliminating the need for labor-intensive manual checks.
- **Providing Instantaneous Feedback:** Real-time updates on parking availability, disseminated through user-friendly applications or dynamic signage, empower drivers with the information needed to swiftly locate vacant spots, thereby minimizing search times and reducing frustration.
- **Alleviating Traffic Congestion:** By streamlining the parking search process, the system contributes to smoother traffic flow, mitigating congestion and enhancing overall urban mobility.

### 1.2.2 Advocating Sustainable Urban Practices

Efficient parking management is not only conducive to improved user experiences but also aligns with broader sustainability objectives:

- **Reducing Fuel Consumption and Emissions:** By minimizing the time spent searching for parking, the system indirectly reduces fuel consumption and associated emissions, contributing to environmental conservation efforts.
- **Enhancing Traffic Flow Efficiency:** Well-managed parking facilities facilitate smoother traffic flow, thereby reducing idle time and congestion-related emissions, ultimately fostering a more sustainable urban environment.

### 1.2.3 Embracing Technological Progress

This project represents an exploration into the real-world applicability of YOLOv4 in solving pressing urban challenges. By developing a bespoke car detection system tailored for parking management, we contribute to the advancement of intelligent transportation systems and smart city initiatives, paving the way for a more connected, efficient, and sustainable urban future.

In essence, the development of a YOLOv4-based car detection system for parking management addresses a critical urban need. It promises to enhance user experiences, mitigate traffic congestion, and foster sustainable urban practices, thereby laying the foundation for smarter, more livable cities of tomorrow.

## 1.3 Dataset

This project's dataset is a comprehensive collection of real-time videos documenting vehicles as they enter parking areas, providing valuable insights into parking lot occupancy. These videos offer a dynamic snapshot of vehicle movement, enabling the accurate counting of vehicles within the parking premises. Within this dataset, meticulous attention is given to essential features such as bounding box sizes and image colors, both of which play crucial roles in the accurate detection and classification of vehicles.

The dataset comprises a substantial volume of images, meticulously divided into distinct subsets for training, validation, and testing purposes. The training dataset consists of 8688 images, providing a rich reservoir of diverse visual data to train the object detection model effectively. Meanwhile, the validation set, comprising 2483 images, serves as a crucial benchmark for evaluating the model's performance during the training process. Finally, the test set, containing 1243 images, offers an independent assessment of the model's efficacy in real-world scenarios, ensuring its generalizability and robustness.

Captured over various driving hours and under different weather conditions, the dataset exhibits a diverse range of environmental contexts, including variations in lighting, weather, and time periods. Despite this diversity, a predominant portion of the dataset comprises images captured during daytime and under clear weather conditions, reflecting the prevalent conditions encountered in typical parking scenarios.

In addition to static images, the dataset also includes real-time videos capturing vehicle movements within the parking area. These videos provide dynamic insights into vehicle ingress and egress, enabling a more comprehensive understanding of parking lot occupancy dynamics. By incorporating both static images and real-time videos, the dataset offers a holistic perspective on parking lot activity, enhancing the object detection model's accuracy and effectiveness.

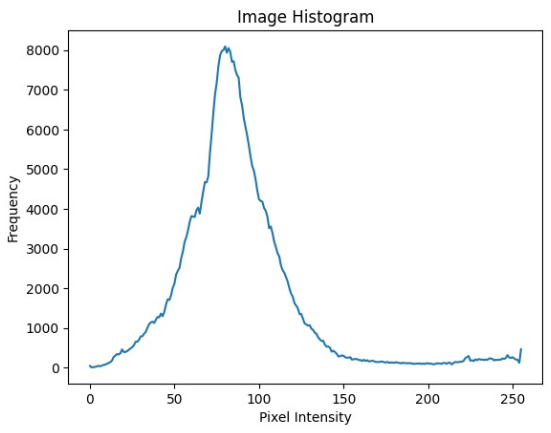


Figure 1.2: Pixel intensity graph of the data

Overall, this dataset serves as a valuable resource for training and evaluating object detection models tailored for parking lot management. Its diverse and comprehensive nature ensures that the resulting models are capable of accurately detecting and counting vehicles under various real-world conditions, ultimately contributing to more efficient and effective parking management solutions.

# Chapter 2

# Review Of Literature

Dodia and S. Kumar [3] compared three versions of the YOLO (You Only Look Once) algorithm, which is a popular and fast object detection model, to detect vehicles in different traffic scenarios. The paper proposes to use a clustering analysis approach to optimize the network structure and the anchor boxes of the YOLO algorithm, and to increase the input image size and the dataset diversity to improve the detection accuracy. The paper evaluates the three versions of the YOLO algorithm on four datasets and shows that the YOLOv3 algorithm outperforms the other two versions in terms of accuracy, precision, recall, and mAP (mean average precision).

R. Patel and P. Meduri [4] proposed a car detection based algorithm for automatic parking space detection, which uses a convolutional neural network (CNN) to detect cars in the parking lot images and then counts the number of vacant and occupied parking spaces. The paper also uses a Kalman filter to track the parking spaces and reduce the false positives and negatives. The paper tests the algorithm on a dataset of 1000 images collected from a parking lot and shows that the algorithm achieves an accuracy of 97.8 and a speed of 0.5 seconds per image.

In [5] authors presents a smart parking system using IoT technology, which uses a camera to capture the images of the parking lot and a Raspberry Pi to process the images and send the data to the cloud. The paper uses an improved YOLOv4 algorithm to detect the vehicles and the parking spaces in the images and a mobile application to display the parking occupancy and the data analytics to the users. The paper demonstrates the system on a prototype parking lot and shows that the system can provide real-time parking information and reduce the energy consumption and the manpower involved.

Table 2.1: Drawbacks of Existing Papers

| **Brief Overview** | **Drawbacks** |
|---|---|
| The[3] project aims to identify the most suitable version of the YOLO (You Only Look Once) algorithm for robust vehicle detection. | It only specifies which algorithm is best for the object detection in present algorithms. |
| The[4] work addresses an important gap in the recent computer vision based artificial intelligence techniques to build smart parking systems. | This project monitors parking lot vacancies on a constant basis, regardless of waiting cars. |
| The [5]system proposes implementation of state-of-the-art Internet of Things (IoT) technology to mold with advanced Honeywell sensors and controllers to obtain a systematic parking system for users. | Using sensors in the IoT system may introduce latency and increase implementation costs compared to a YOLO-based system. |

In [6] author address the challenges posed by urbanization and growing populations. Traditional parking management methods are often costly and inefficient. The researchers propose a vision-based approach using existing streetlight cameras. By leveraging deep learning techniques, they accurately identify vacant parking spaces, reducing search time for drivers. This costeffective solution contributes to smart city development and efficient urban planning.

In [7], the authors have used a thermal infrared camera to reliably capture objects in low visibility and high contrast conditions like during sunrise, sunsets, at night, etc. In this paper, an object detection algorithm for AVs is developed by the direct external infrared parameter correction algorithm between the thermal infrared cameras and LiDAR sensors. A 3D calibration target was manufactured to match the characteristics of the thermal IR cameras and LiDAR sensor, and extrinsic parameters were calibrated.

The authors of [8] listed various two stage and one stage object detection methods. Two stage methods explained are: RCNN, Fast RCNN, Faster R-CNN, and Mask R-CNN. One stage methods are: YOLO, Single Shot Detector (SSD), RetinaNet, FCOS, CornerNet and ExtremeNet and Transformers. The models based on the YOLOR algorithm are the most performant for 2D object detection. The limitations discussed involve the use of cameras, like occlusions and low resolution in certain scenarios. Cameras and LIDAR data can be fused to overcome limitations and improve detection robustness and accuracy. There are cameras involved which leads to occlusion and low resolution limitations in certain scenarios. Camera and LiDAR data can be fused to overcome these limitations and improve detection robustness and accuracy.

# Chapter 3

# You Only Look Once (YOLO) for Object Detection

This chapter will go over the fundamentals of visual object identification, including the object detection model, YOLO's neural network, deep learning, and computer vision. Following that, we'll go into the complexities of the YOLOv4 architecture, providing a thorough grasp of its operating foundation and features.

## 3.1 You Only Look Once (YOLO)

Object detection is framed as a regression issue to distinct bounding boxes and related class probabilities in the You Only Look Once (YOLO) algorithm. A single neural network is used to predict bounding boxes and class probabilities from images in a single assessment .[9]

### 3.1.1 The YOLO Detection System

The YOLO (You Only Look Once) detection system represents a significant advancement in object detection algorithms, renowned for its exceptional speed and accuracy. Unlike conventional methods that typically involve multi-stage processes such as region proposal, feature extraction, and classification, YOLO operates on a single pass through a convolutional neural network (CNN). This single-shot approach enables YOLO to directly predict bounding boxes and class probabilities for multiple objects in an image, streamlining the detection process and significantly reducing computational complexity. Figure 3.1 illustrates how the YOLO detection algorithm classifies three items based on their confidence levels.

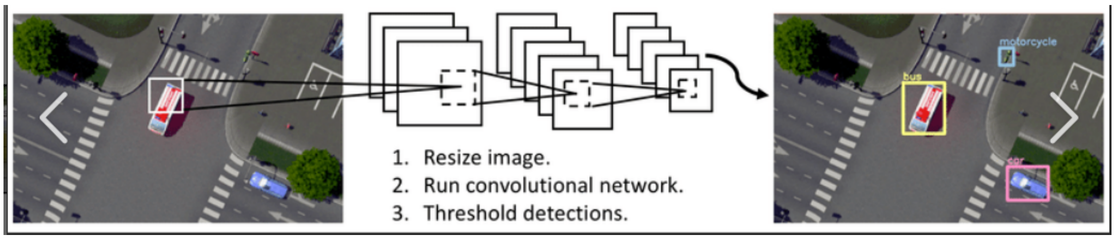


Figure 3.1: The YOLO Detection System.

One of the key features of YOLO is its ability to perform simultaneous prediction of bounding boxes and class probabilities, allowing it to detect multiple objects of different classes within a single image frame. This capability is made possible by the integration of convolutional layers within the neural network architecture, which enable the model to extract spatial features and semantic information from the input image.

Furthermore, YOLO employs a technique known as non-maximum suppression (NMS) to filter redundant bounding boxes and retain only the most confident detections. This post-processing step helps improve detection accuracy by eliminating duplicate detections and reducing false positives. By effectively suppressing overlapping bounding boxes with lower confidence scores, YOLO ensures that only the most relevant detections are retained for further analysis.[10]

Another advantage of YOLO is its scalability and adaptability to diverse object detection tasks. The algorithm can be trained on large datasets containing a wide range of object classes, making it suitable for applications across various domains, including autonomous driving, surveillance, and image analysis. Additionally, the YOLO algorithm has undergone several iterations, with each iteration introducing improvements in speed, accuracy, and robustness. The latest iteration, YOLOv4, represents the culmination of these advancements, offering state-of-the-art performance in object detection tasks.

The YOLO detection system offers a powerful and efficient solution for object detection tasks, characterized by its speed, accuracy, and scalability. By leveraging a single-pass CNN architecture and innovative techniques such as non-maximum suppression, YOLO revolutionizes the field of computer vision and enables real-time detection of objects in diverse applications.

## 3.2 YOLOv4

### 3.2.1 Modern Object Detector

Normally, a modern detector has two parts: a backbone and a head. The backbone is pretrained on ImageNet, and the head is for predicting classes and bounding boxes. Some backbones for GPU platforms include VGG, ResNet, and DenseNet, while for CPU platforms include SqueezeNet, MobileNet, and ShuffleNet. Object detectors are categorized into two kinds based on the head: a one-stage object detector and a two-stage object detector. Examples of two-stage object detector models are R-CNN series, Faster R-CNN, R-FCN, and Libra R-CNN, while examples of one-stage object detector models are YOLO, SSD, and RetinaNet. Nowadays, some layers between the backbone and the head are inserted to develop detectors which are called the neck of an object detector. These layers collect feature maps from different stages. Feature Pyramid Network (FPN), Path Aggregation Network (PANet), BiFPN, and NAS-FPN can be used as the neck of a network .

Before going to the main architecture of YOLOv4, some essential concepts are described below.

### 3.2.2 Bag of Freebies

A bag of freebies is a training method that can increase the accuracy of the object detector without affecting the inference cost. In object detection, a bag of freebies usually means data augmentation. Data augmentation increases the variability of the input images, making the detection model more robust to images obtained from different environments. Two common data augmentation methods are photometric distortions and geometric distortions. In photometric distortion, the brightness, contrast, hue, saturation, and noise of an image are adjusted. In geometric distortion, random scaling, cropping, flipping, and rotating are done .

### 3.2.3 Bag of Specials

Bag of specials includes some plugin modules and post-processing methods that increase the inference cost slightly but improve the accuracy of object detection on a large scale. These plugin modules enlarge the receptive field, introduce attention mechanisms, or strengthen feature integration capability. Commonly used modules are Spatial Pyramid Pooling (SPP), Atrous Spatial Pyramid Pooling (ASPP), and Receptive Field Block (RFB). Post-processing is for screening model prediction results .

### 3.2.4 Cross Stage Partial DenseNet

In Cross Stage Partial DenseNet (CSPDenseNet), the feature map of the base layer is partitioned into two and merged through a cross-stage hierarchy to make the computation efficient. A partial dense block and a partial transition layer are two components of a stage of CSPDenseNet. The partial dense blocks increase the gradient path, balance computation of each layer, and reduce memory traffic. The partial transition layer maximizes the difference of gradient combination by using the strategy of truncating the gradient flow. It prevents layers from learning duplicate gradient information. In a partial dense block, the feature maps of the base layer are split into two parts. $X_0 = [X_0', X_0'']$. $X_0''$ is directly linked to the end of the stage, and $X_0'$ goes through a dense block. The output of dense layers, $[X_0', X_1, ..., X_k]$, undergoes a transition layer. Then, the output of this transition layer, $XT$, is concatenated with $X_0''$ and undergoes another transition layer, then output $XU$ is generated .

## 3.3 YOLOv4 Architecture

YOLOv4 is a fully convolutional network with a total of 110 convolution layers, consisting of 66 layers with a size of 1 × 1 and 44 layers with a size of 3 × 3. The input layer comprises a 3 × 3 convolution layer with 32 filters. It accepts input images of size 416 × 416 with 3 channels (RGB) in this thesis. The output layer is a 1 × 1 convolution layer with a stride and padding size of 1, featuring 33 filters. YOLOv4 utilizes CSPDarknet53 as the backbone and incorporates SPP and PAN for the Neck, while the head of YOLOv3 is used in YOLOv4. Mini-batch gradient descent with momentum is employed for optimization. CSPDarknet-53 effectively extracts deep features from the input images. SPP enhances the receptive field efficiently, while PANet extracts features across multiple scales, and the heads are responsible for object detection. A linear activation function is used for the final layer.[11]

The Bag of Freebies (BoF) for the backbone includes Mosaic data augmentation and DropBlock regularization. Bag of Specials (BoS) for the backbone consists of Mish activation and Cross-stage partial connections (CSP). BoF for the detector includes CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, using multiple anchors for single ground truth, optimal hyperparameters, and random training shapes. Bag of Specials for the detector encompasses Mish activation, SPP-block, PAN-block, and DIoU-NMS.

Mosaic mixes 4 training images, and CutMix mixes 2 input images to enable the detection of objects outside their normal context. Various methods of data augmentation such as MixUp and Blur are employed. Self-Adversarial Training (SAT) involves altering the original image, followed by training the neural network to detect objects in this modified image. Cross mini-Batch Normalization (CmBN) collects statistics only between mini-batches within a single batch.

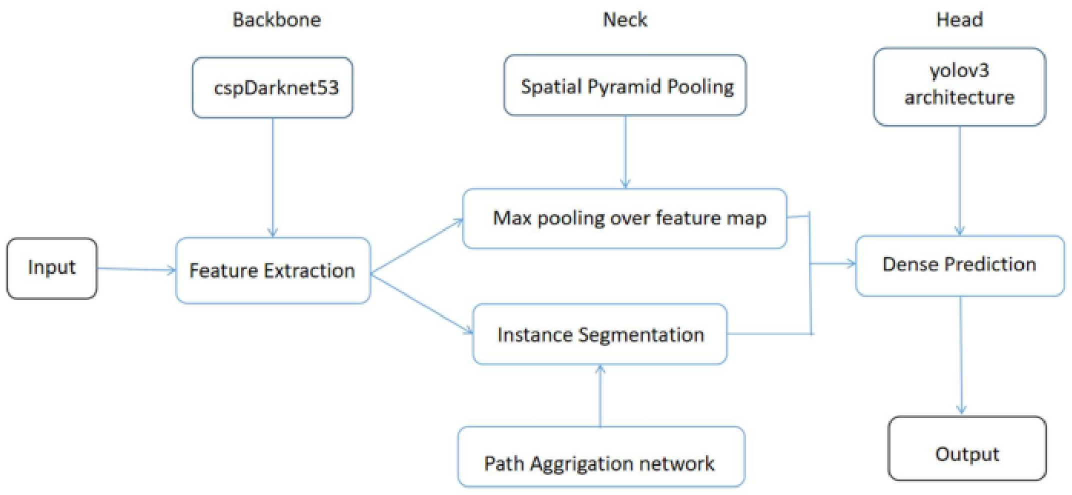
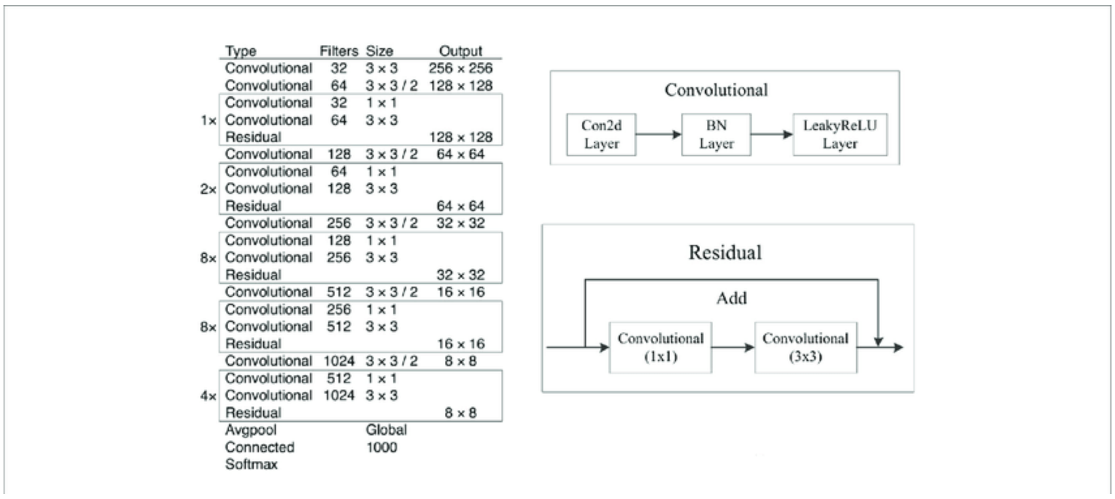


Figure 3.2: Yolov4 Architercture

### 3.3.1 CSPDarknet53

CSPDarknet53 is the combination of Darknet53 and CSPDenseNet. It is a convolutional neural network and backbone for object detection. Darknet53 has 53 convolutional layers. It contains 53 convolution layers of sizes $1 \times 1$ and $3 \times 3$. Among them, 29 convolution layers are $3 \times 3$. Every convolution layer is connected with a batch normalization (BN) layer and a Mish activation layer. CSPDarknet53 is used as the backbone of YOLOv4 and Darknet53 for YOLOv3 .



| | Type | Filters | Size | Output |
|---|---|---|---|---|
| | Convolutional | 32 | 3 × 3 | 256 × 256 |
| | Convolutional | 64 | 3 × 3 / 2 | 128 × 128 |
| 1× | Convolutional | 32 | 1 × 1 | |
| | Convolutional | 64 | 3 × 3 | |
| | Residual | | | 128 × 128 |
| | Convolutional | 128 | 3 × 3 / 2 | 64 × 64 |
| 2× | Convolutional | 64 | 1 × 1 | |
| | Convolutional | 128 | 3 × 3 | |
| | Residual | | | 64 × 64 |
| | Convolutional | 256 | 3 × 3 / 2 | 32 × 32 |
| 8× | Convolutional | 128 | 1 × 1 | |
| | Convolutional | 256 | 3 × 3 | |
| | Residual | | | 32 × 32 |
| | Convolutional | 512 | 3 × 3 / 2 | 16 × 16 |
| 8× | Convolutional | 256 | 1 × 1 | |
| | Convolutional | 512 | 3 × 3 | |
| | Residual | | | 16 × 16 |
| | Convolutional | 1024 | 3 × 3 / 2 | 8 × 8 |
| 4× | Convolutional | 512 | 1 × 1 | |
| | Convolutional | 1024 | 3 × 3 | |
| | Residual | | | 8 × 8 |
| | Avgpool | | Global | |
| | Connected | | 1000 | |
| | Softmax | | | |

Figure 3.3: Darknet53

### 3.3.2 Spatial Pyramid Pooling

Spatial Pyramid Pooling (SPP) helps the network to use different sizes of images instead of requiring a fixed-size input image in the traditional system. It uses max-pooling layers of different sizes of 5, 9, and 13 to pool the features and generates fixed-size outputs. YOLOv4 uses an SPP block after CSPDarknet53 to extract important features from the backbone .

### 3.3.3 Path Aggregation Network

Path Aggregation Network (PANet), a feature pyramid network, is used in YOLOv4 to boost information for segmentation. It uses top-down and bottom-up approaches to extract important features across multiple scales from the backbone classifier to enhance performance. Output of PANet is fed to detection heads. PANet is replaced by concatenation in YOLOv4, and Leaky-ReLU is used instead of ReLU .[12]

### 3.3.4 Head

The system predicts 4 coordinates $(tx, ty, tw, th)$ for the bounding boxes. If $(cx, cy)$ is the coordinator of the top left corner of the bounding box and $(wgt, hgt)$ and $(w, h)$ is the height and the width of the ground truth bounding box and predicted bounding box, respectively. Then the predictions would be:

$$bx = \sigma(tx) + cx \tag{3.1}$$

$$by = \sigma(ty) + cy \tag{3.2}$$

$$w = wgte^t w \tag{3.3}$$

$$h = hgte^t h \tag{3.4}$$

Here, $(bx, by)$ represents the center coordinates of the predicted box, $\sigma(tx)$ and $\sigma(ty)$ are two functions of $tx$ and $ty$, respectively .

The boxes predict in their contained classes, and binary cross-entropy is used as a loss. The system predicts 3 different scales (13 × 13, 26 × 26, 52 × 52 for input size 416 × 416) of boxes to fuse and interact with feature maps of different scales to detect objects of different sizes. A total of 8, 16, and 32 strides are used to convert a 416 × 416 image into 52 × 52, 26 × 26, and 13 × 13 images, respectively. 52 × 52 is used to detect small objects, 26 × 26 is used to detect medium objects, and 13 × 13 is used to detect large objects. YOLOv3 and YOLOv4 use the same head in their prediction layers .[11, 12]

# Chapter 4

# Proposed work with it's Methodology

Vehicle detection and counting are becoming increasingly important in the field of highway management. However, due to the different sizes of vehicles, their detection remains a challenge that directly affects the accuracy of vehicle counts. To address this issue, this report proposes a vision based YOLO algorithm with the use of deep convolutional network's (CNNs) strong ability to learn image features

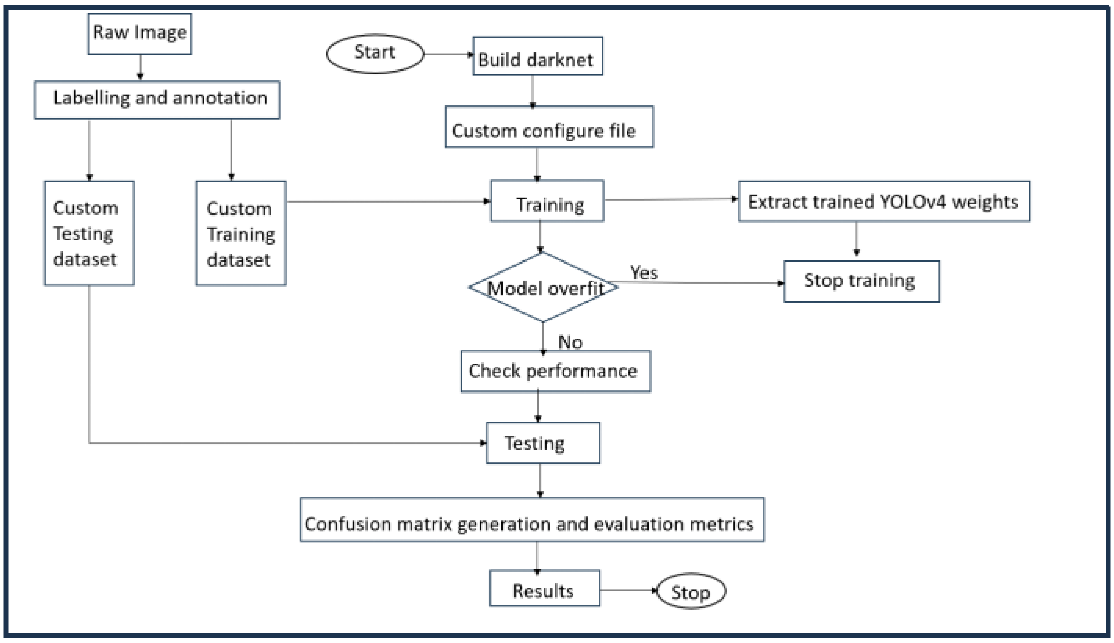


Figure 4.1: Block diagram of proposed method

The overall proposed plan is compromised into the following steps :

1. Pre processing step.

2. Vehicle Detection.

3. Performance Evalution.

## 4.1 Pre Processing Step

- When capturing images in a dark environment, the color and texture information degrades. The degradation of image features change the performance of vacant parking space detection. An image is a matrix composed of a pixel. Image processing is to manipulate the pixel matrix.
- **Image Enhancement** : There will be a lot of noise in the image because the camera will be disturbed by many factors when taking photos. It will interfere with the identification of parking spaces. The region of interest in the image is enhanced to reduce noise interference and improve the accuracy of recognition. That is to preserve the details of the image and suppress the noise as much as possible, which is necessary to filter. The filtering effect like image augmentation, sheering effect will directly affect the subsequent processing and recognition.
- **Convolutional neural network** : CNNs are specialized kinds of neural networks that use convolution instead of general matrix multiplication in at least one of the layers. The YOLO v4 network uses CSPDarkNet-53 which has 53 convolutional layers of sizes 1×1 and 3×3. Among them, 29 convolution layers are 3×3 .This is the backbone for extracting features from the input images which is later discussed in YOLO architecture.

## 4.2 Training on Custom Dataset

A major part of our project involved training the model on a custom dataset, in addition to examining current parking availability detection methods and implementing car detection using the YOLOv4 architecture. Accurately representing real-world scenarios required gathering and annotating a wide range of parking lot photographs. Each image has been painstakingly tagged to indicate the presence and placement of cars within the parking slots.[9, 11]

To guarantee reliable model performance and generalization, we then divided the dataset into subsets for testing, validation, and training. We adjusted the YOLOv4 model parameters through successive training sessions in order to maximize detection accuracy and reduce false positives. Furthermore, we utilized data augmentation methods including rotation, scaling, and flipping to expand the dataset and improve the model's adaptability to diverse environmental circumstances.

Our goal was to fine-tune the YOLOv4-based automobile identification system to precisely predict parking availability in real-world scenarios by carefully selecting and training on a customized dataset. In addition to improving the model's performance, this rigorous training procedure demonstrated our dedication to creating a solid and trustworthy parking management system.

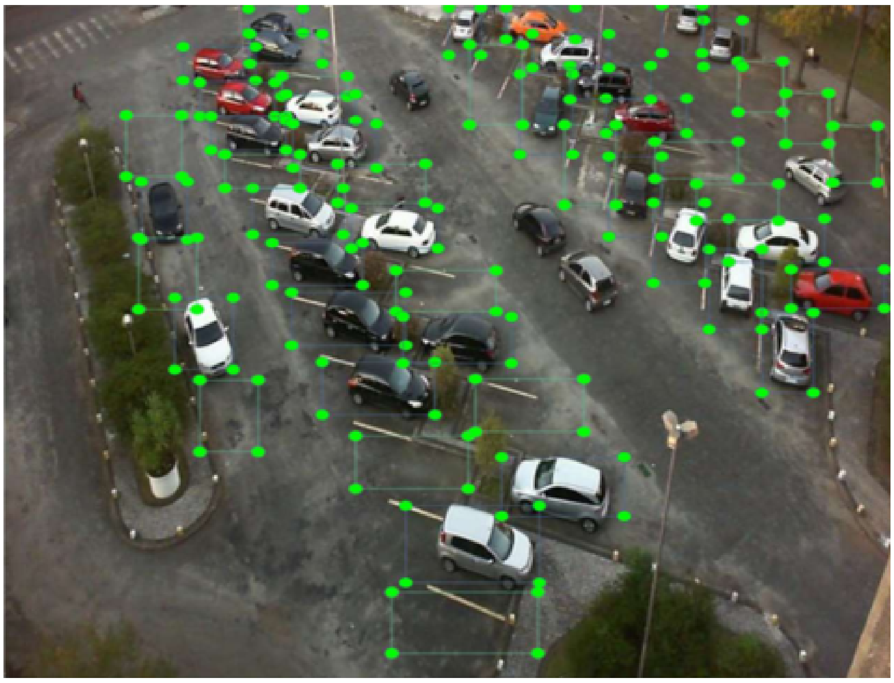
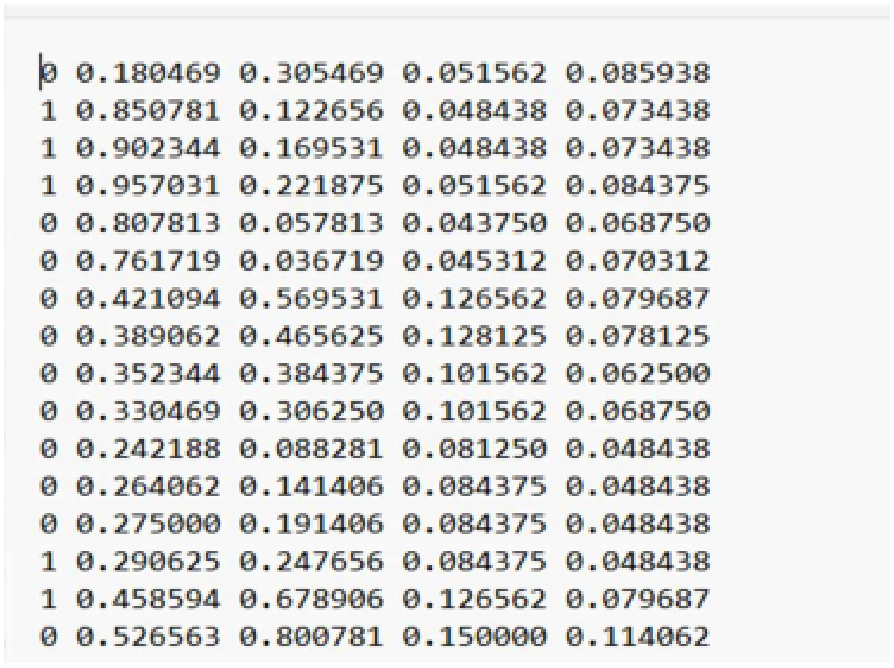


Figure 4.2: Annoted Image



```
0 0.180469 0.305469 0.051562 0.085938
1 0.850781 0.122656 0.048438 0.073438
1 0.902344 0.169531 0.048438 0.073438
1 0.957031 0.221875 0.051562 0.084375
0 0.807813 0.057813 0.043750 0.068750
0 0.761719 0.036719 0.045312 0.070312
0 0.421094 0.569531 0.126562 0.079687
0 0.389062 0.465625 0.128125 0.078125
0 0.352344 0.384375 0.101562 0.062500
0 0.330469 0.306250 0.101562 0.068750
0 0.242188 0.088281 0.081250 0.048438
0 0.264062 0.141406 0.084375 0.048438
0 0.275000 0.191406 0.084375 0.048438
1 0.290625 0.247656 0.084375 0.048438
1 0.458594 0.678906 0.126562 0.079687
0 0.526563 0.800781 0.150000 0.114062
```

Figure 4.3: Yolov4 type annotation

We collected a range of photos showing the desired parking lot under various circumstances, such as varied lighting, weather, and viewing positions. Every one of these photos was manually annotated, with labels designating empty spaces and bounding boxes placed around cars with care. To aid in the creation and assessment of the model, the dataset was later divided into subsets for training, validation, and testing.

We began our training approach with pre-trained YOLOv4 weights as a fundamental starting point in order to accelerate convergence. The model was skillfully adjusted to take into account the unique subtleties of the parking lot layout, the automobiles that are there, and the changing lighting conditions through repeated fine-tuning on our bespoke dataset. This methodical approach made sure that the model was optimized and adapted for precise automobile detection in the assigned parking area.

## 4.3 Vehicle detection

Vehicle detection entails detecting all vehicles that enter a specific Region of Interest ROI. To accurately localize an object's position, modern object detection methods use rectangular-shaped, horizontal/vertical bounding boxes drawn over it.

- In YOLO , input images are divided into an S x S grid.

- If the center of an object falls into a grid cell , that grid cell is liable for detecting that object .Each grid cell predicts bounding boxes and confidence scores for those boxes by using the features from whole image.
- These confidence scores reflect the probability that there's an object in box and accuracy of prediction of an object class.
- The confidence score is defined as :

$$\text{Confidence} = \Pr(\text{class}_1|\text{object}) \times \Pr(\text{object}) \times \text{IoU}_{\text{thresh}} \tag{4.1}$$

  where $0 \leq \Pr(\text{object}) \leq 1$.
- Here , $p_r(\text{object})$means the probability of being an object in the grid cell.
- $p_r(class_i|\text{object})$ means that the probability of a specific object presents in the cell given that the cell contains an object.
- IoU Intersection over Union (IoU) is the area of overlap between the predicted bounding box and the target bounding box, divided by the area of their union (Figure 6.3). It is an evaluation metric that is used to evaluate the accuracy of object detection.

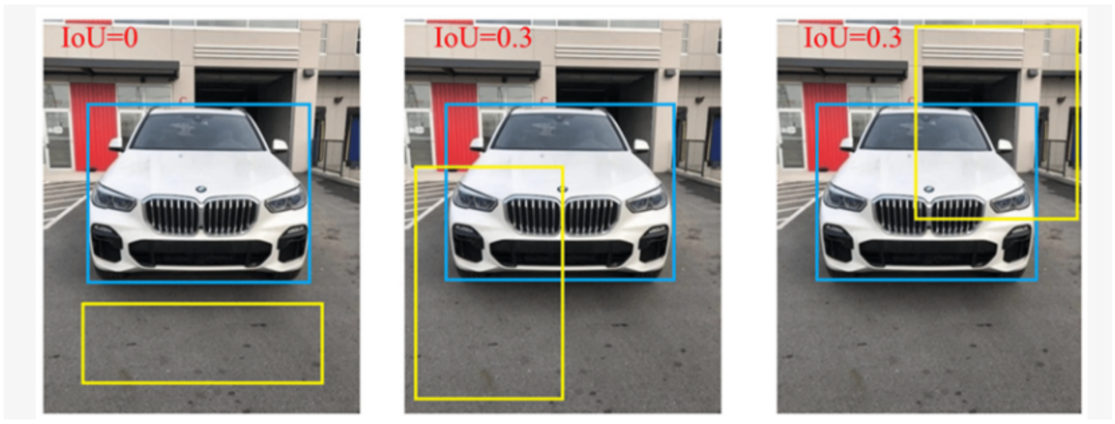


Figure 4.4: Intersection over Union (IoU) metric for bounding boxes.

## 4.4 Car Tracking

The objective of the car tracking algorithm is to differentiate between stationary cars and moving cars in a sequential processing of frames from the parking lot. This algorithm operates by comparing the bounding boxes for cars detected in the current frame with a list of tracked cars, referred to as potential parking spots. The comparison is based on the Intersection over Union (IoU) Ratio, which quantifies the overlap between two bounding boxes (Equation 1). When the IoU exceeds a predefined threshold (IoU Threshold), the bounding boxes are considered a match.

The algorithm considers three main scenarios:

1. **New Car Entry:** When a new car enters the frame (Fig. 1), its bounding box does not match any existing potential parking spots, so it's added to the list.

2. **Stationary Car:** For a stationary car (Fig. 4.5), its bounding box matches one in the potential parking spots list, indicating that it's been in that location for some time. A counter is incremented for this spot, and if it surpasses a predefined threshold (Frame Threshold), the location is marked as a parking spot.

3. **Moving Car:** A moving car (Fig. 4.6) will have its bounding box in the potential parking spots list but fails to match any bounding boxes in the current frame before reaching the frame threshold. In this case, it's removed from the list.

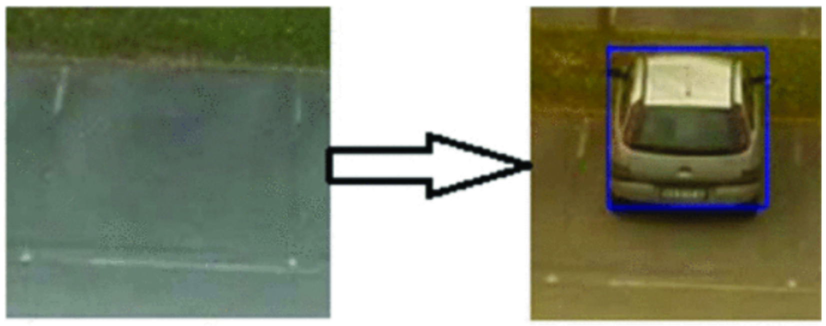


Figure 4.5: New car

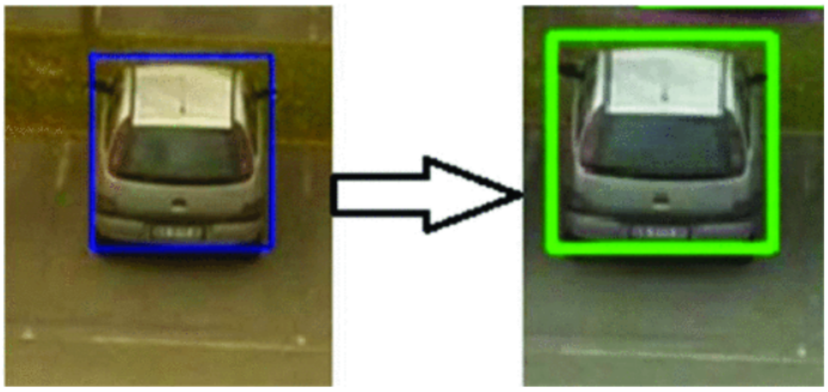


Figure 4.6: Moving Car

Careful selection of IoU threshold and Frame threshold is crucial for distinguishing between moving and parked cars. A higher IoU threshold increases sensitivity to small changes in bounding box location, potentially leading to redundant detections. However, it should be high enough to ignore the intersection of adjacent parking spots. Empirical findings suggest an IoU threshold between 0.35-0.5 yields optimal results.

The frame threshold is chosen based on frame rate and the desired static time of a car. For instance, if the interval between frames is 5 minutes and a car needs to be stationary for 20 minutes to be considered parked, the frame threshold would be 4. This algorithm's effectiveness relies on appropriately tuning these thresholds to balance sensitivity and specificity in car detection and tracking.

# Chapter 5

# Evaluation Parameters

**Evaluation Methodology for Object Detection**

1. **Classification Metrics**

   For classification, the sample data is categorized into four different categories based on the original and predicted data:

   - **True Positive (TP)**: Correct detections where the intersection over union (IoU) is greater than or equal to the threshold.

   - **False Positive (FP)**: Incorrect detections where the IoU is less than the threshold.

   - **False Negative (FN)**: Ground truths that were not detected, including incorrect detections where the IoU with the ground truth is above the threshold.

   - **True Negative (TN)**: Not applicable in object detection tasks due to the impracticality of calculating all possible bounding boxes that were correctly not detected.

2. **Formulas for Precision, Recall, and F1-Score**

   Precision and recall are calculated using the following formulas:

$$P = \frac{TP}{TP + FP} \tag{5.1}$$

$$R = \frac{TP}{TP + FN} \tag{5.2}$$

$$F1 = \frac{2 \times P \times R}{P + R} \tag{5.3}$$

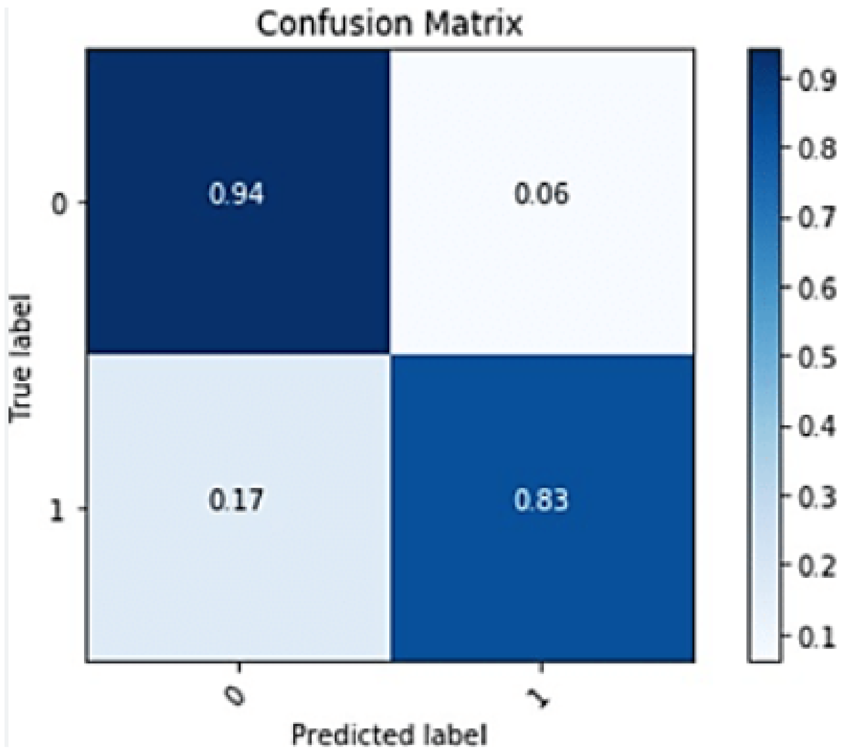


Figure 5.1: Confusion Matrix

Table 5.1: Evaluation of the models on the test data

| **Metric** | **YOLOv4** |
|---|---|
| Average IoU | 83.10% |
| Precision | 85.70% |
| Recall | 75.00% |
| F1-score | 80.20% |

3. **Model Performance Assessment**

   The model's performance is evaluated based on its ability to predict bounding boxes, their locations, and contained object classes. Key metrics such as mAP and IoU values are utilized to measure the model's accuracy and effectiveness in object detection tasks.

4. **Considerations and Limitations**

   It's essential to acknowledge the limitations of TN in object detection tasks and focus on precision, recall, and F1-score for evaluating the model's efficacy comprehensively.

By incorporating these evaluation methodologies and metrics, a thorough assessment of the object detection model's performance is conducted, providing insights into its capabilities and areas for improvement.

# Chapter 6

# Results

As an initial phase of our project, we employed a pre-trained YOLOv4 model to conduct car detection. The results, depicted in Figure 6.1, illustrate successful detection of cars of various colors.

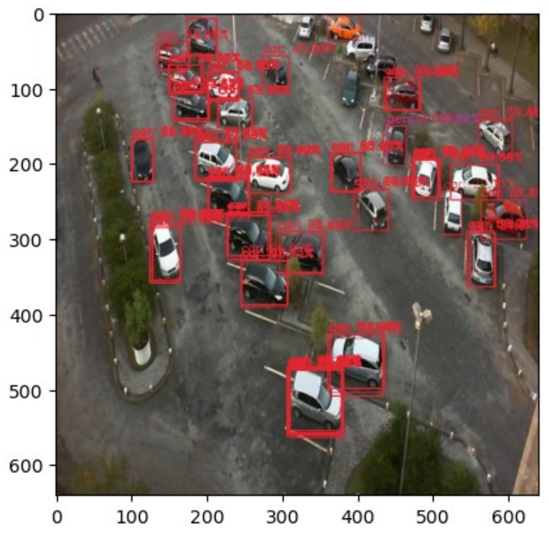


Figure 6.1: Cars detected using Yolov4 trained with Coco dataset

Subsequently, we proceeded to train the model using a custom dataset. The dataset was annotated to classify empty parking lots as class 0 and parking lots occupied by vehicles as class 1, across various weather conditions. Figure 6.2 illustrates the successful detection of both cars and empty parking lots.

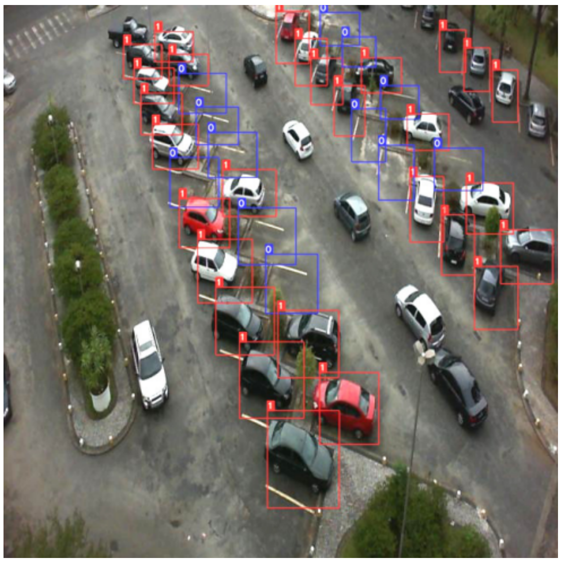


Figure 6.2: Car and Empty lot detection

Additionally, we conducted training on random road videos, where we implemented a line-crossing approach. We assumed that when a car crosses this line, it enters the parking area. We then counted the number of cars that crossed the line. If the count is not zero, indicating the presence of cars, the parking lot detection process initiates, providing the number of empty parking lots available.

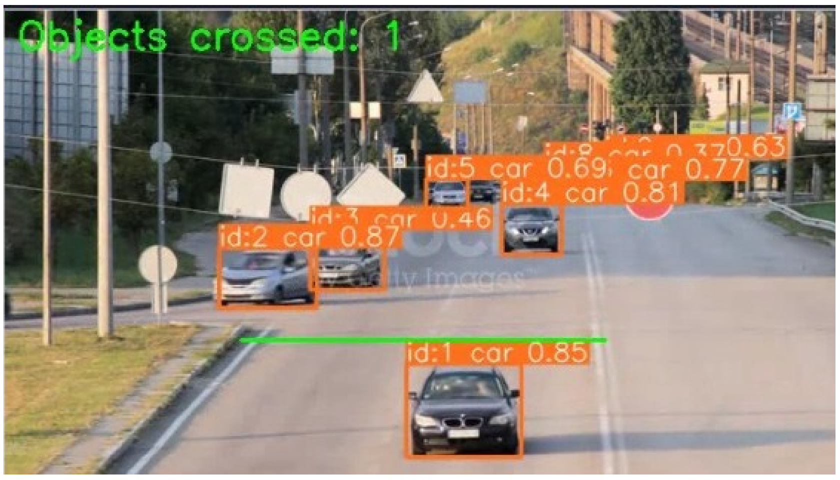


Figure 6.3: Counting the cars entering the parking area

# Chapter 7

# Conclusion and Future scope

## 7.1 Conclusion

In conclusion, the implementation of YOLOv4 for parking lot availability prediction represents a pivotal step towards revolutionizing urban transportation infrastructure management. Through the development and deployment of a sophisticated car detection system, capable of swiftly and accurately identifying vehicle presence within parking lots, this project addresses the longstanding challenges associated with parking management in densely populated urban environments. By providing real-time insights into parking availability, the system not only enhances the overall user experience but also plays a crucial role in mitigating traffic congestion, reducing environmental impact, and fostering sustainable urban mobility practices.

The successful utilization of YOLOv4 underscores its efficacy as a cutting-edge object detection algorithm, capable of delivering unparalleled performance in terms of speed, accuracy, and scalability. By leveraging state-of-the-art deep learning techniques, this project demonstrates the potential of AI-driven solutions to tackle complex urban challenges and enhance the quality of life for urban residents and visitors alike.

## 7.2 Future Scope

Looking forward, the potential for further advancements and enhancements in this domain is vast. There are several avenues for extending the scope and impact of the YOLOv4-based parking availability prediction system:

1. **Fine-Tuning and Optimization:** Continued refinement of the system's algorithms and parameters can further improve detection accuracy, particularly for small or occluded objects, and enhance overall performance.

2. **Integration with Smart City Initiatives:** Seamless integration with broader smart city initiatives, such as intelligent transportation systems and urban planning frameworks, can facilitate holistic urban management and optimization.

3. **User-Centric Design and Accessibility:** Prioritizing user-centric design principles and ensuring accessibility for all users can enhance the usability and adoption of the parking availability prediction system among diverse urban stakeholders.

4. **Collaborative Partnerships and Data Sharing:** Collaborating with local authorities, parking operators, and technology partners to share data and insights can enable more comprehensive and accurate predictions, leading to more efficient parking management strategies.

5. **Continuous Monitoring and Evaluation:** Ongoing monitoring and evaluation of the system's performance in real-world settings are essential to identify areas for improvement and ensure long-term effectiveness and sustainability.

In essence, the successful deployment of YOLOv4 for parking lot availability prediction underscores the transformative potential of AI-driven technologies in addressing complex urban challenges. By embracing innovation, collaboration, and a user-centric approach, we can pave the way for smarter, more efficient, and more sustainable cities of the future.

# Bibliography

[1] Cui Gao, Qiang Cai, and Shaofeng Ming. Yolov4 object detection algorithm with efficient channel attention mechanism. In *2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pages 1764–1770, 2020.

[2] Raj Patel and Praveen Meduri. Car detection based algorithm for automatic parking space detection. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1418–1423, 2020.

[3] A. Dodia and S. Kumar. A comparison of yolo based vehicle detection algorithms. In *2023 International Conference on Artificial Intelligence and Applications (ICAIA) Alliance Technology Conference (ATCON-1)*, pages 1–6, 2023.

[4] R. Patel and P. Meduri. Car detection based algorithm for automatic parking space detection. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1418–1423, 2020.

[5] D. Ashok, A. Tiwari, and V. Jirge. Smart parking system using iot technology. In *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pages 1–7, 2020.

[6] L.-C. Chen, R.-K. Sheu, W.-Y. Peng, J.-H. Wu, and C.-H. Tseng. Video-based parking occupancy detection for smart control system. *Applied Sciences*, 10:1079, 2020.

[7] T. Agrawal and S. Urolagin. Multi-angle parking detection system using mask r-cnn. In *Proceedings of the 2020 2nd International Conference on Big Data En-*

*gineering and Technology*, pages 76–80. Association for Computing Machinery, 2020.

[8] Mazin Hnewa and Hayder Radha. Object detection under rainy conditions for autonomous vehicles. 2020.

[9] Hyung–Jun Jin and Hyong–Suk Kim. A study on paprika disease detection with yolov4 model using a customed pre-training method. In *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, pages 1615–1618, 2021.

[10] P C Manojkumar, Lakshmi Sutha Kumar, and B Jayanthi. Performance comparison of real time object detection techniques with yolov4. In *2023 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT)*, pages 1–6, 2023.

[11] C. Dewi, RC. Chen, and X. et al. Jiang. Deep convolutional neural network for enhancing traffic sign recognition developed on yolo v4. *Multimedia Tools and Applications*, 81:37821–37845, 2022.

[12] S. Ali, A. Siddique, H. F. Ateş, and B. K. Güntürk. Improved yolov4 for aerial object detection. In *2021 29th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4, 2021.

# Appendices

# Appendix A

# Code Attachments

## A.1 Yolov4 code

```
import cv2
import numpy as np

img=cv2.imread('image-path')
img_width=img.shape[1]
img_height=img.shape[0]
img_blob=cv2.dnn.blobFromImage(img, 1/255, (416,416), swapRB=True,
    crop=False)

labels = ["car","empty"]
colors = ["255,0,0","0,255,255"]
colors = [np.array(color.split(",")).astype("int") for color in
    colors]
colors = np.array(colors)

model = cv2.dnn.readNetFromDarknet('yolov4.cfg-path',''yolov4.weights
    - path')
scalefactor = 1.0/255.0
new_size = (416, 416)
blob = cv2.dnn.blobFromImage(img, scalefactor, new_size, swapRB=True,
     crop=False)

class_labels_path = r"YOLOV4.coco.names-path"
class_labels = open(class_labels_path).read().strip().split("\n")
class_labels
# Red, lime, Blue, Yellow, Magenta,black,white,silver,grey,maroon,
    green,navy
class_colors = ["255,0,0","0,255,0","0,0,255","255,255,0","255,0,
    255","0,0,0","255,255,255","192,192,192","128,128,128","128,0,0",
"0,128,0","0,0,128"]


class_colors = [np.array(every_color.split(",")).astype("int") for
    every_color in class_colors]
class_colors = np.array(class_colors)
class_colors = np.tile(class_colors,(16,1))

def colored(r, g, b, text):
    return "\033[38;2;{};{};{}m{} \033[38;2;255;255;255m".format(r, g
    , b,text)

for i in range(16):
    line = ""
    for j in range(12):
        class_id = i*12 + j
        class_id_str = str(class_id)
        text = "class" + class_id_str
        colored_text = colored(class_colors[class_id][0],
    class_colors[class_id][1], class_colors[class_id][2], text)
        line += colored_text
    print(line)

# or select the colors randomly
class_colors = np.random.randint(0, 255, size=(len(class_labels), 3),
     dtype="uint8")


yolo_model = cv2.dnn.readNetFromDarknet('yolov4.cfg-path',''yolov4.
    weights- path')
model_layers = yolo_model.getLayerNames()
print("number of network components: " + str(len(model_layers)))

output_layers = [model_layers[model_layer - 1] for model_layer in
    yolo_model.getUnconnectedOutLayers()]
print(output_layers)

yolo_model.setInput(blob)
obj_detections_in_layers = yolo_model.forward(output_layers)
print("number of sets of detections: " + str(len(
    obj_detections_in_layers)))


class_ids_list = []
boxes_list = []
confidences_list = []
img_height = img.shape[0]
img_width = img.shape[1]
confidence_threshold=0.15

# loop over each output layer
for object_detections_in_single_layer in obj_detections_in_layers:
# loop over the detections in each layer
    for object_detection in object_detections_in_single_layer:
      # get the confidence scores of all objects detected with the
    bounding box
      prediction_scores = object_detection[5:]
      # consider the highest score being associated with the winning
    class
      # get the class ID from the index of the highest score
      predicted_class_id = np.argmax(prediction_scores)

      # get the prediction confidence
      prediction_confidence = prediction_scores[predicted_class_id]


      # consider object detections with confidence score higher than
    threshold
```

```python
      if prediction_confidence > confidence_threshold:
        # get the predicted label
        predicted_class_label = class_labels[predicted_class_id]
        # compute the bounding box coordinates scaled for the input image
        bounding_box = object_detection[0:4] * np.array([img_width, img_height, img_width, img_height])
        (box_center_x_pt, box_center_y_pt, box_width, box_height) = bounding_box.astype("int")
        start_x_pt = max(0, int(box_center_x_pt - (box_width / 2)))
        start_y_pt = max(0, int(box_center_y_pt - (box_height / 2)))

        # update the 3 lists for nms processing
        # - confidence is needed as a float
        # - the bbox info has the openCV Rect format
        class_ids_list.append(predicted_class_id)
        confidences_list.append(float(prediction_confidence))
        boxes_list.append([int(start_x_pt), int(start_y_pt), int(box_width), int(box_height)])

# i=0
# j=0
max_ids = cv2.dnn.NMSBoxes(boxes_list, confidences_list, 0.50, 0.4)
m=len(max_ids)
x=0
for max_id in max_ids:
    max_class_id = max_id
    box=boxes_list[max_class_id]

    start_x = box[0]
    start_y = box[1]
    box_width = box[2]
    box_height = box[3]

    predicted_id =class_ids_list[max_class_id]
    label = class_labels[predicted_id]
    confidence = confidences_list[max_class_id]


    end_x = start_x + box_width
    end_y = start_y + box_height

    box_color = class_colors[predicted_id]
    box_color = [int(each) for each in box_color]

    if label == "car":
        i=i+1
    else:
        j=j+1

    label = "{}: {:.2f}%".format(label, confidence*100)
    x+= confidence*100

    print("Predicted object {}".format(label))
    cv2.rectangle(img,(start_x,start_y),(end_x,end_y),box_color,1)
test_imgz = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.imshow(test_imgz)
plt.show()
avg=x/m
print(avg)


print("\nNUMBER OF EMPTY PARKING SPACES DETECTED".format(j))
print("NUMBER OF VEHICLES DETECTED={}".format(i))
```