A MAJOR PROJECT REPORT
ON
SMART CAR PARKING SYSTEM USING ARTIFICIAL INTELLIGENCE

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award

BACHELOR'S DEGREE IN ELECTRONICS & ELECTRICAL ENGINEERING

By

ANKIT SARKAR - 1707178
SAIKAT CHAKRABORTY - 1707218
SOUVIK MAL - 1707228
SUMAN MONDAL - 1707237
ABHISEK PATI - 1707251

UNDER THE GUIDANCE OF :
PROF: PRIYA DAS

SCHOOL OF ELECTRONICS ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA – 751024
APRIL 2021

KIIT Deemed to be University

School of Electronics Engineering Bhubaneswar, ODISHA 751024

CERTIFICATE

This is to certify that the project entitled

"SMART CAR PARKING SYSTEM USING ARTIFICIAL INTELLIGENCE"

Submitted by:

ANKIT SARKAR - 1707178
SAIKAT CHAKRABORTY - 1707218
SOUVIK MAL - 1707228
SUMAN MONDAL - 1707237
ABHISEK PATI - 1707251

is a record of Bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Electronics and Electrical Engineering) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2020-2021, under our guidance.

Date:2/04/2021

Prof. Priya Das

ACKNOWLEDGEMENTS

We are profoundly grateful to Prof. PRIYA DAS for her expert guidance and encouragement throughout to see that this project rights its target since its commencement to its completion. The work is a team effort minus which the completion of this project was not possible.

Ankit Sarkar
Saikat Chakraborty
Souvik Mal
Suman Mondal
Abhisek Pati

ABSTRACT

Smart car parking system is very challenging problem nowadays.Everybody face difficulties while parking car in very crowd or busy places like hospital,Mall,offices etc. The main motive of the smart car parking system are many : 1) Parking function are “benifit-driving” example: unsupervised parking and valet parking. 2) Driving functions are “pain-driven”: available despite icy/foggy/dark/ wet condition etc. Participants clearly prioritize in two use cases 1) Automated parking maneuver with driver in vehicle 2) Automated driving in heavy rain. It also increases the business value in the market and increase the comfort of the customer or vehicle user. Besides comfort it also help to manage the time for every people in busy life. Hence,this is the very great idea to implement in the busy world. So in this project the demo of the smart car parking system project has been shown by using artificial intelligence Masked RCNN.

Keywords --Video processing , Car parking images , Convolutional neural network, Image processing, Computer vision, Sending sms by Twilio ,Masked RCNN.

CONTENTS

Topic
Page No

1. INTRODUCTION

Smart Parking is a parking solution that can include in-ground Smart Parking sensors, cameras or counting sensors. These devices are usually embedded into parking spots or positioned next to them to detect whether parking bays are free or occupied. This happens through real-time data collection. The data is then transmitted to a smart parking mobile application or website, which communicates the availability to its users. Some companies also offer other in-app information, such as parking prices and locations. This gives you the possibility to explore every parking option available to you.

Smart Parking and its Smart Parking Sensors can be seen as a part of smart cities. These smart cities are cities that are driven by an IT infrastructure and by using this infrastructure, cities can enhance the quality of life and improve economic development for its inhabitants. Becoming a smart city can be a good way to collect historical data in a relatively easy way. By collecting this data, cities can analyze how processes, like parking can be optimized.

As a result of using Smart Parking, people who are looking to find a parking spot will find it in the most efficient way possible and companies or municipalities can optimize their parking territories. It also makes cities more livable, safer and less congested.

Advantages of Smart Parking for Drivers:-

Optimizing the driving experience: using a Smart Parking system saves a lot of time for drivers since they know where to find a vacant parking spot. The amount of time you spend while looking for a parking spot will be minimized. By using the Parkeagle technology of the Smart Parking sensors, you will be able to find the parking spot you are looking for, without having to browse through the streets.

Advantages for Cities

1) Less pollution: Smart Parking contributes to a cleaner environment. Reducing the time that is necessary to find a parking spot will reduce the amount of fuel that is used when looking for a parking space. This makes the process of finding a parking spot contribute to less pollution, which is beneficial for everyone.

2) The space of a municipality will be utilized more efficiently: because Smart Parking sensors transmit live-data, drivers will have a real-time overview of the occupancy of parking

bays. This means that free spots can be filled quicker, which will reduce the time that a parking spot is empty.

3) Safety: The use of Smart Parking Sensors can optimize safety within cities. As a result of placing, for instance, on-ground sensors on parking bays, people will not be as stressed as when they are looking for parking spaces. Because these people will know where they are going, they can simply navigate to their parking spot and they will not have to stress out about it.

4) Real-time parking analytics for cities: Parking space will become intelligent by use of the smart parking sensors on the parking bays. This means that as a city you're able to see historical data which is stored and you're able to make data driven decision and predictions based on the parking sensor data.

In general, Smart Parking solutions, such as sensors, give municipalities and companies the opportunity to make parking a more fluid and efficient process. Furthermore, it saves people a great amount of time, money, and reduces the frustration that a person might have when wanting to find a parking spot.

1.1 MOTIVATION

The increase in city traffic is one of the major effects of population growth especially in urban areas. Due to this searching for a vacant parking area during peak hours is not only time-consuming but also results in wastage of fuel. The drivers keep searching for suitable parking lot which leads to increase in traffic. Increasing volume of vehicular exhaust creates a negative impact on the environment. The drivers keep searching for suitable parking lot which leads to increase in traffic. Increasing volume of vehicular exhaust creates a negative impact on the environment. Hence reservation-based smart parking has become the need of the day

2 INTRODUCTION

2.1.Machine Learning:

It is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so that Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks.

Performing machine learning involves creating a model, which is trained on some training data and then can process additional data to make predictions. Various types of models have been used and researched for machine learning systems.

- Artificial neural network
- Decision trees
- Genetic analysis
- Regression

But in order to complete our project we have used the concepts of ARTIFICIAL NEURAL NETWORK.

2.2. Deep Learning

It is part of a broader family of machine learning methods based on artificial neural networks with representation learning.Learning can be supervised, semi-supervised or unsupervised. Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bio informatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains. Specifically, neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analog.

Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces. Most modern deep learning models are based on artificial neural networks, specifically, Convolutional Neural Networks (CNN)s, although they can also include propositional formulas or latent

variables organized layer-wise in deep generative models such as the nodes in deep belief networks and deep Boltzmann machines

In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level on its own.Deep learning has various applications in real word they are :

Automatic speech recognition

Image recognition

Visual art processing

Medical image analysis

Image restoration

Automatic recommendation system

Military uses

Artificial neural network

Natural language processing (the voice of Google Assistant

2.3. Flowchart of Deep Learning algorithm

A deep neural network provides state-of-the-art accuracy in many tasks, from object detection to speech recognition. They can learn automatically, without predefined knowledge explicitly coded by the programmers.

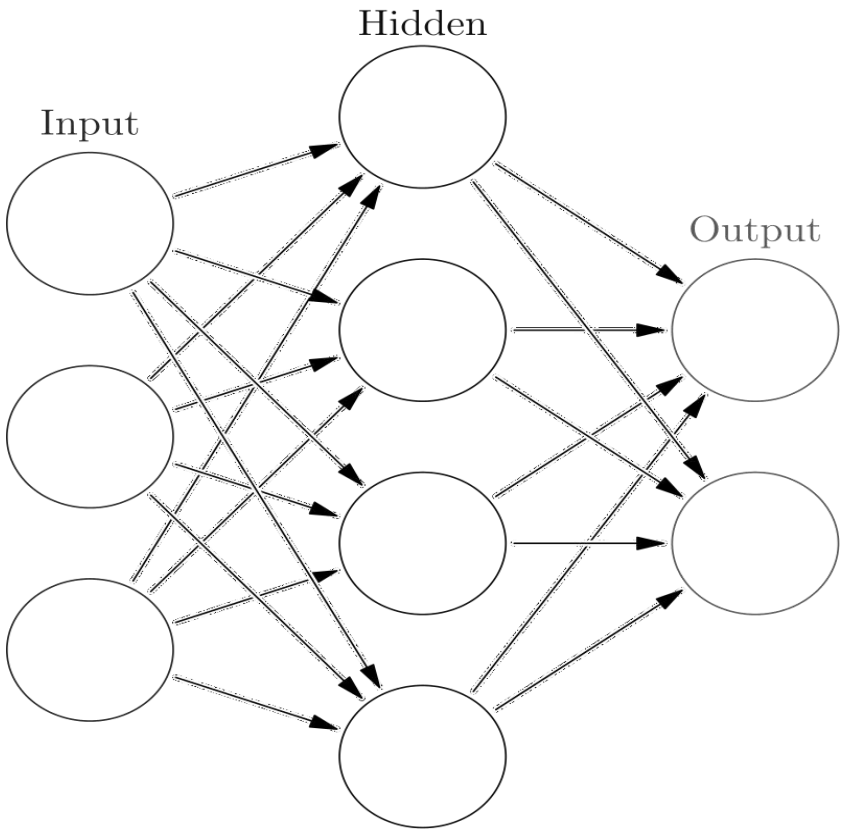
To grasp the idea of deep learning, imagine a family, with an infant and parents. The toddler points objects with his little finger and always says the word 'cat.' As its parents are concerned about his education, they keep telling him 'Yes, that is a cat' or 'No, that is not a cat.' The infant persists in pointing objects but becomes more accurate with 'cats.' The little kid, deep down, does not know why he can say it is a cat or not. He has just learned how to hierarchies complex features coming up with a cat by looking at the pet overall and continue to focus on details such as the tails or the nose before to make up his mind.

A neural network works quite the same. Each layer represents a deeper level of knowledge, i.e., the hierarchy of knowledge. A neural network with four layers will learn more complex feature than with that with two layers.

2.3. Artificial neural network:

These are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.An ANN is a model based on a collection of connected units or nodes called "artificial neurons", which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit information, a "signal", from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers multiple times.

Artificial neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.Deep learning consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing. Some successful applications of deep learning are computer vision and speech recognition.

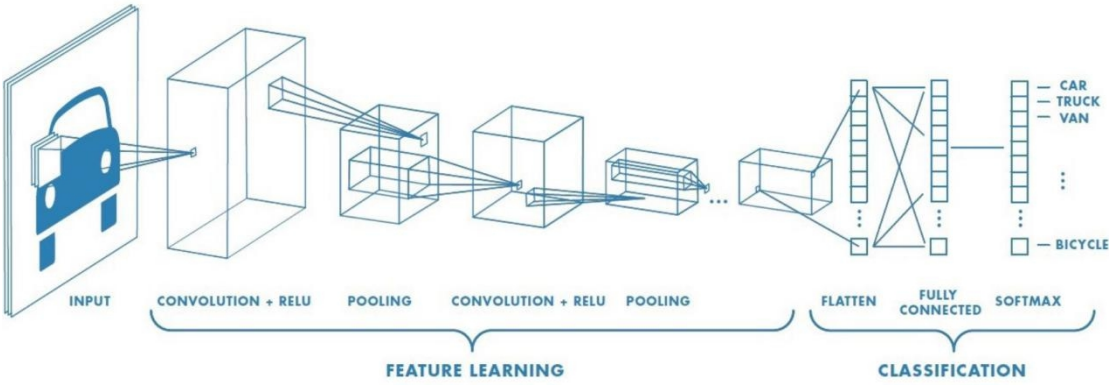


3. OBJECTIVE

Objective first is to detect the number of cars present in the parking lot from surveillance camera by using computer vision technology . After that region of interest (ROI) will be calculated by using Masked RCNN . Hence we can easily identify the parking lot and the car. Once the parking lot is detected whether the car is parked or not in the particular parking area then this message will be send to the car ,by this car driver will easily able to know the update of the parking space and can be easily parked in the particular area.

4. CNN SYSTEM DESIGN

In convolutional neural network there are having four type of layer convolutional layer . maxpooling layer , flattening layer and fully connected layer. So in convolutional layer image is convoluted with some filter and kernel dimension then the resultant image is scaled down to minmax scaling as convolutional layer is generally applied in the image for edge detection so scaling is done for better edge detection. Now that scaling image is passed to the rectifier linear unit activation function as image itself contains some non linear elememts. However the convolution image dimension is less than from the original image so some of the information get missing so after convolution padding is done to maintain the same dimension of the image.After that resulltant image is passed through the maxpooling layers due to time invariant that means sometimes input images having different form like rotated images with different angle ,zooming, shrinking, horizontal ,vertical flip in that case model cannot recognise the image while prediction so maxpooling filter of certain dimension is applied with the resultant images. After that images is flattend from sparse matrix as the flattend matrix is finally passed to the fully connected layer for training and learning.



5.Video processing using opencv

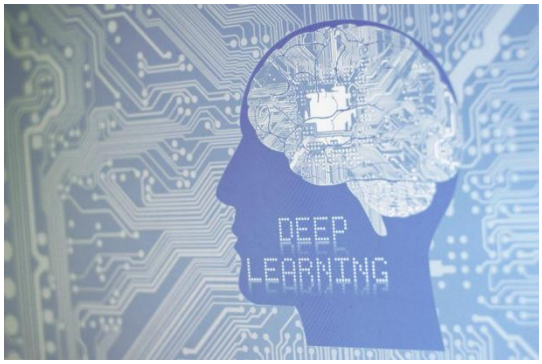
When we detect some object through video then some steps are need to be followed. That can be only possible by video processing opencv. For every video for detection we need to remove noise or high frequency component by using low pass filter. Then that video background should be blur by using high pass filter. After that video need to be masked that means setting the pixel value according to the given matrix value.To detect the portion or edges of the car will be detected by using canny edge detector. So,there are many steps like this are involved in the video processing ,further morphological operation are need for proper visualization.

6.PROBLEM STATEMENT

Sometimes it's very difficult to park car in the proper parking due to crowd and busy area. Let's take one example, you have gone to office/hospital/Mall and you are in emergency situation and you have to park a car in crowd parking area and you have no idea whether parking space are available or not ,once you enter into parking lot and you find no parking area are available so it's take your time in your emergency situation. It's really happen with everyone. So, here in this project we have tried to implement smart car parking system which will help to manage your time and make car parking easier. In this project we will detect the whether the car parking space available or not , if available that message will sent to the respective car driver who are searching for parking space along with parking area number so that driver can easily parked a car in a proper place without any hindrance. Lot of research has been already done with this project.So here we have try to implement the same.

7.Technologies used

7.1 Deep Learning



Lets consider a example a new born baby he is not aware about anything in this world . First he is trained for walk then he is trained to talk then he admitted to school for study . So if we see the whole life cycle for human being every human being is need to learn something everyday for survival without learning they can't do or tell anything . In human being brain , neuron is the main thing for learning and taking dicision so each and every decison taken by human being is decide by the brain . So deep learning is the technology where we teach our computer through artificially created neuron that called multi layer perceptron so after lots of training computer can predict ,detect or can tell everything itself which we have teach or trained it. Let's take another example during our exam time if we take a revison for particular subject for one or two times we can't write properly during our exam time but if we make revison many times we can write properly during our exam . Similary if we train the system very less time we will get very less accuracy while prediction. So deep learning is very much relevant to the human beings.

7.2 Digital Image Processing

Digital image processing is a technique where we can apply various operation in the images to process the image or analysis to find out the particular

information from the images . In digital image processing first preprocessing step is applied like resizing, colour contrasting,normalization , histogram processing , colour converstion. In post processing steps includes thresholding, segmenation , morphological operation these steps are applied to extract some of the information from the images.

8.Libraries Used

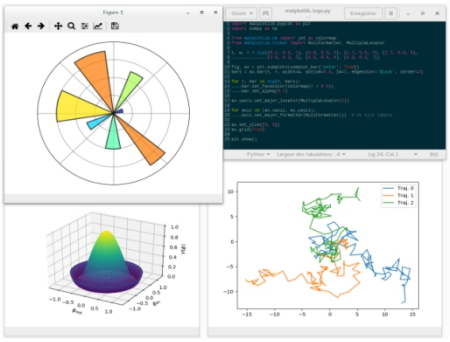
8.1 OpenCV

Opencv is the very important library for computer vison and image processing techniques. This library allows to import the images for operation . With this library we can do various operation like resizing, colour contrasting,normalization , histogram processing , colour converstion. In post processing steps includes thresholding, segmenation , morphological operation these steps are applied to extract some of the information from the images.

8.2 Numpy

NumPy is a library for the Python programming language, it mainly allow for various mathematical operation like matrix multiplication ,vector addition , vector substraction , to generate random number and many more . So this is very important tool for machine learning and deep learning.

8.3 Matplotlib



Matplotlib is a very useful library in machine and deep learning as well for plotting purpose. It provides various types of plotting like histogram ,bar plot,scatter plot,linear plot , pie plot etc. It also used for multivariate , univariate ,bivariate ploting .In our project we can visualize the images for each and every processing and analysis by matplotlib library.

8.4 KERAS

Keras is a high level neural network API it run around the tensor computation.Through keras library we can create artifical neural network,convolutional neural network and recurrent neural network. Keras library mainly used to create neural network framework.

8.5 Twilio

Twilio is python API that used to send sms to the provided number from the twilio services. This library is used in our project to send sms to the user whether parking area is available or not in the particular parking space.

9.Tools Used:-

Anaconda Python 3.6 environment

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012.Package versions in Anaconda are managed by the package management system conda.

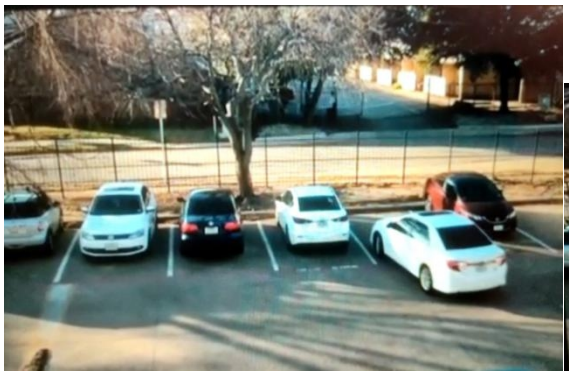
Google colab:-

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

9.1 DATASETS:

Here we are using video dataset , where it has been shown that a car left its respective parking area and area made available ,that video is processed and undergo masked RCNN and computer vision algorithm by this whether parking space is available or not can able to know other driver wanted to park his car in the parking area.






10. PROJECT PLANNING FOR SMART CAR PARKING SYSTEM.

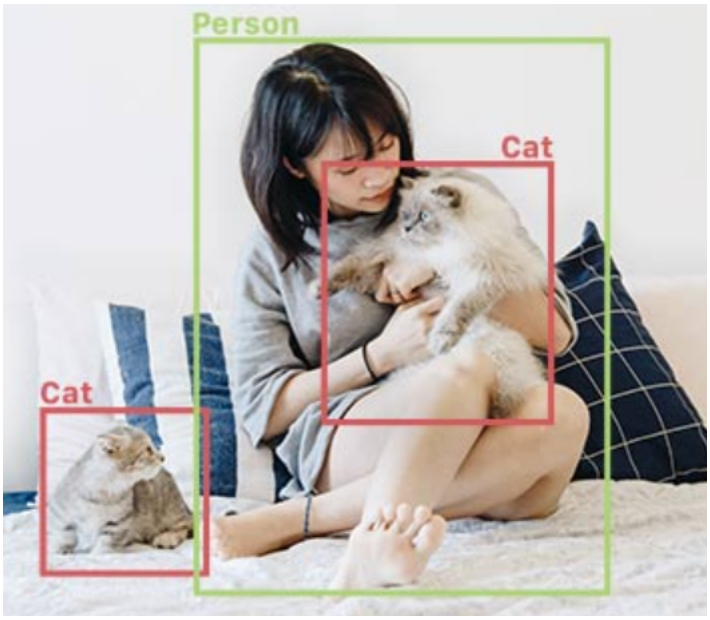
So here in this project first is to detect the number of cars present in the parking lot from surveillance camera by using computer vision technology . After that region of interest (ROI) will be calculated by using Masked RCNN . Hence we can easily identify the parking lot and the car. Once the parking lot is detected whether the car is parked or not in the particular parking area then this message will be send to the car ,by this car driver will easily able to know the update of the parking space and can be easily parked in the particular area.

10.Object Detection

Introduction

Object detection is a computer vision technique that permits us to spot and locate objects in a picture or video. With this sort of identification and localization, object detection is being used to count objects during a scene and determine and track their precise locations, all while accurately labeling them.

Imagine, for instance, a picture that contains two cats and someone. Object detection allows us to directly classify the categories of things found while also locating instances of them within the image.



But how does object detection actually work? What are the various approaches, what are its major potential benefits and limitations, and the way might you utilize in your business?

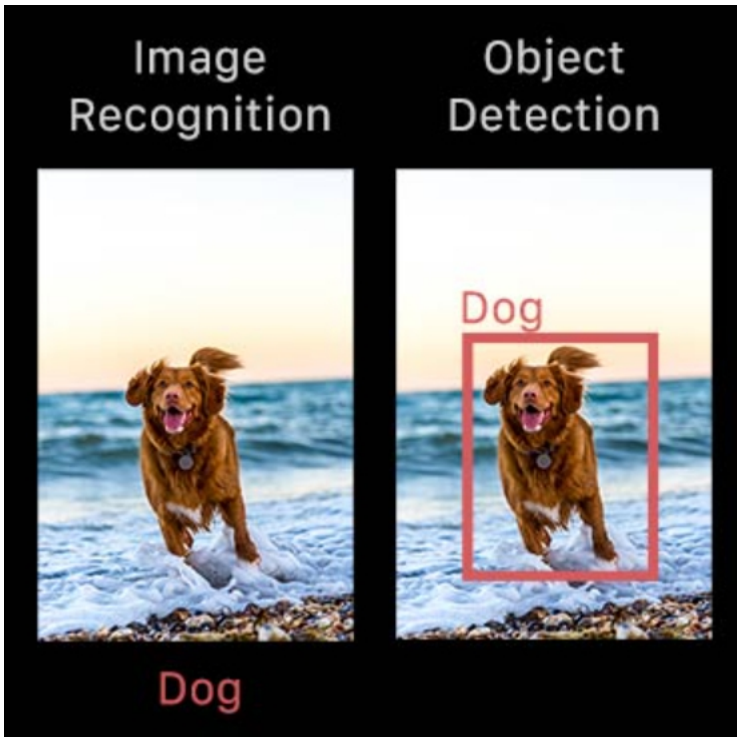
Part 1: Object detection – the basics

Object detection is basically a computer vision technique that works to spot and locate objects within a picture or video. Specifically, object detection draws bounding boxes around these detected objects, which permit us to locate where said objects are in (or how they move through) a given scene.

Object detection is often confused with image recognition, so before we proceed, it's important that we clarify the distinctions between them.

Image recognition assigns a label to an image. An image of a dog receives the label "dog". An image of the two dogs, still receives the label "dog". Object detection, on the opposite hand, draws a box around each dog and labels the box as "dog". The model predicts where each object is and which label should be applied. In this way, object detection provides more information about a picture than recognition.

Here's an example of the how this distinction looks in practice:



Modes and types of object detection

In more traditional ML-based approaches, computer vision techniques are being used to take a look at various features of an image, like the colour histogram or edges, to spot groups of pixels which will belong to an object. These features are then fed into a regression model which predicts the location of the object together with its label.

In the other hand, deep learning-based approaches employ convolutional neural networks (CNNs) to perform end-to-end, unsupervised object detection, within which features don't have to be defined and extracted separately.

Because deep learning methods became the state-of-the-art approaches to object detection, these are the techniques we'll be that focusing in for the needs of this guide.

Why is object detection important?

Object detection is inextricably linked to other similar computer vision techniques like image recognition and image segmentation, in this it helps us understand and analyze scenes in images or video.

But there are several important differences.As Image recognition only

outputs a class label for an identified object, and where as image segmentation creates a pixel-level understanding of a scene's elements. What separates object detection from these other tasks is its unique ability to locate objects within a picture or video. This then allows us to count so track those objects.

Given these key distinctions and object detection's unique capabilities, we will see how it will be applied in a very number of ways:

a. It is used in Crowd counting
b. It is used in Anomaly detection
c. It is used in Face detection
d. It is used in Self-driving cars
e. It is used in Video surveillance

Of course, this isn't an exhaustive list, but it includes a number of the first ways within which object detection is shaping our future.

Part 2: How does object detection work?

Now that we all know a small amount about what object detection is, the distinctions between the differing types of object detection, and what it will be used for, let's explore in additional depth how it actually works.

In this section, we'll study several deep learning-based approaches to object detection and assess their advantages and limitations. even as a reminder—for the needs of this overview, we will be viewing the approaches that are used in neural networks, which became the state-of-the-art methods for object detection.

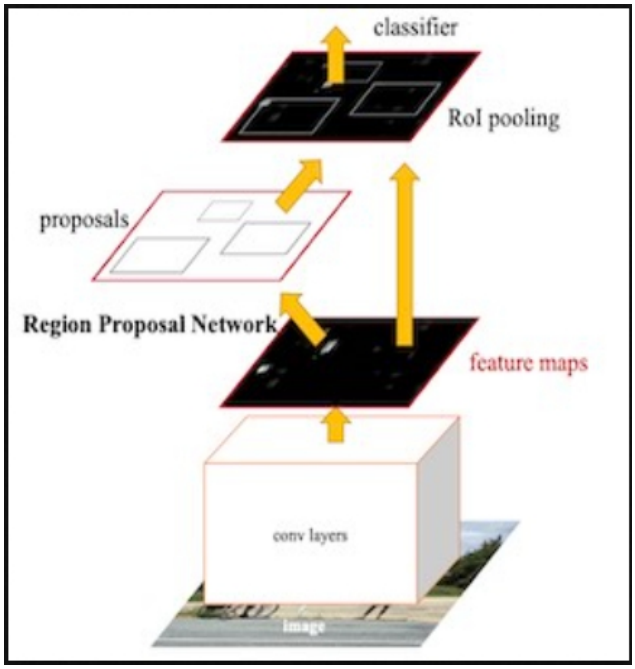
Basic structure

Deep learning-based object detection models usually have 2 components. An encoder takes an image as an input and runs it through a series of blocks and layers that are learn to extract statistical features which are used to locate and label objects. Outputs from the encoder are then passed to a decoder, which predicts bounding boxes and labels for each and every object.

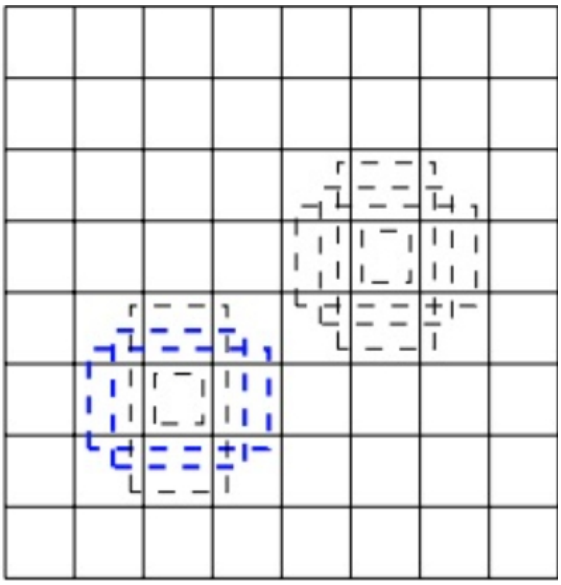
The simplest decoder is the pure regressor. The regressor is connected to the output of the encoder and predicts the location and size of each and every bounding box directly. The output of the model is that the X, Y coordinate pair for the given object and its extent within the image.Though simple, this kind of model is proscribed or limited. You wish to specify the amount of boxes sooner than time. If your image has two dogs, but your model was only designed to detect one object, one will go unlabeled. However, if you

recognize the amount of objects you are needed to predict in each image sooner than time, pure regressor-based models is also a decent option.

An extension of the regressor approach is basically a region proposal network. While using this decoder, the model proposes regions of an images where it believes that an object might reside. The pixels belonging to those regions are then fed into a classification sub network to work out or to determine a label (or reject the proposal). It then runs the pixels which containing those regions through a classification network. The good thing about this method it could be a more accurate or rather is more accurate, flexible model that may propose arbitrary numbers of regions that will contain a bounding box. The added accuracy, though, comes at the value of computational efficiency.



Single shot detectors (SSDs) seek a middle ground. instead of employing a sub network to propose regions, SSDs depend upon a group of predetermined regions. A grid of anchor points is laid over the input image, and at each anchor point, boxes of multiple shapes and sizes function regions. for every box at each anchor point, the model outputs a prediction of whether or not an object exists within the region and modifications to the box's location and size to create it fit the object more closely. Because there are multiple boxes at each anchor point and anchor points is also close, SSDs produce many potential detections that overlap. Post-processing must be applied to SSD outputs so as to prune away most of those predictions and pick the simplest one. the foremost popular post-processing technique is understood as non-maximum suppression.



Finally, a note on accuracy. Object detectors output the location and label for every object, but how can we understand how well the model is doing? For an object's location, the foremost commonly-used metric is intersection-over-union (IOU). Given two bounding boxes, now we compute the area of the intersection and divide by the area of the union.Now This value ranges from 0 (no interaction) to 1 (perfectly overlapping). For labels, a straightforward "percent correct" are often or mostly used.

Model architecture overview

R-CNN, Faster R-CNN, Mask R-CNN

A number of popular object detection models belongs to the R-CNN family. Short for region convolutional neural network, these architectures are supported the region proposal structure discussed above. Over the years, they have become both more accurate and more computationally efficient. Mask R-CNN is that the latest iteration, developed by researchers at Facebook, and it makes a decent start line for server-side object detection models

YOLO, MobileNet + SSD, SqueezeDet

There are variety of models that belong to the one shot detector family. Most of the difference between these variants are their encoders and also the specific configuration of predetermined anchors. MobileNet + SSD models

feature a MobileNet-based encoder, SqueezeDet borrows the SqueezeNet encoder, and therefore the YOLO model features its own convolutional architecture. SSDs make a great choices for models destined for mobile or embedded devices.

CenterNet

More recently, researchers have developed object detection models that do away with the requirement for region proposals entirely. CenterNet treats objects as the single points, predicting the X, Y coordinates of an object's center and its extent (height and width). This system has proven both more efficient and accurate than SSD or R-CNN approaches.

How object detection works on the edge

If your use case requires that object detection add real-time, without internet connectivity, or on private data, you may be considering running your object detection model directly on a foothold device sort of a transportable like a mobile phone or IoT board.

In those cases, you'll have to choose specific model architectures to create sure everything runs smoothly on these lower power devices. Here are some tips and tricks to make sure your models are ready for edge deployment:

a. Prune your network to incorporate or include fewer convolution blocks. Most papers use network architectures that aren't constrained by compute or memory resources. This results in networks with much more layers and parameters than are required to come up with acceptable predictions.

b. Add a width multiplier to your model so you'll be able to adjust the quantity or number of parameters in your network to satisfy your computation and memory constraints. the quantity of filters during a convolution layer, for instance, greatly impacts the general size of your model. Many papers and open-source implementations will treat this number as a set constant, but most of those models were never intended for mobile use. Adding a parameter that multiplies the bottom number of filters by a relentless or constant fraction allows you to modulate the model architecture to suit the constraints of your device. for a few tasks, you'll create much, much smaller networks that perform even as well as large ones.

c. Shrink models with quantization, but watch out or beware for accuracy drops. Quantizing model weights can save a bunch of space, often

reducing the scale of a model by an element of 4 or more. However, accuracy will suffer. ensure you test quantized models rigorously to work out if they meet your needs.

d. Input and output sizes is smaller than you think! If you're designing a photograph organization app, it's tempting to think that your object detection model must be ready to accept full resolution photos as an input. In most of the cases, edge devices won't have nearly enough processing power to handle this. Instead, it's common to train the object detection models at modest resolutions, then downscale input images at runtime.

Part 3: Use cases and applications

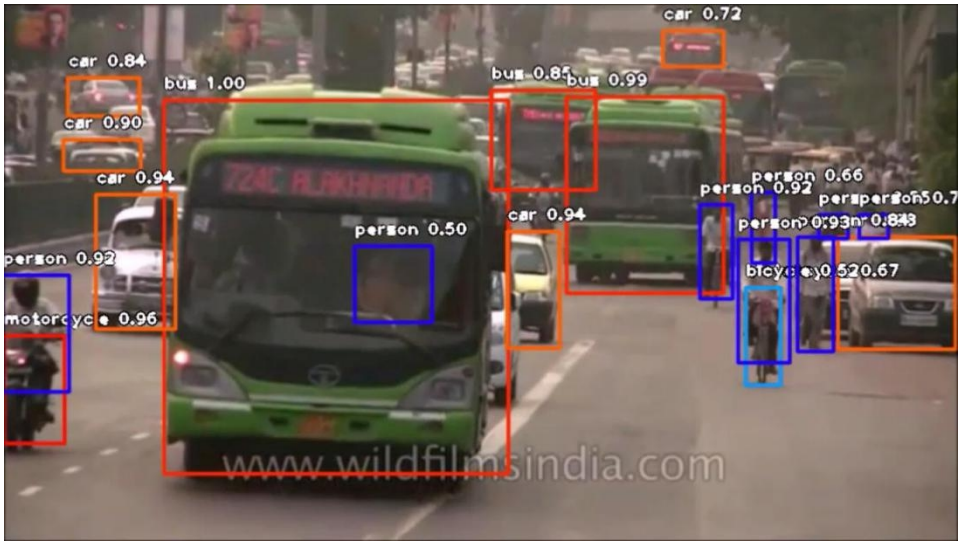
In this section, we'll provide an summary of real-world use cases for object detection.

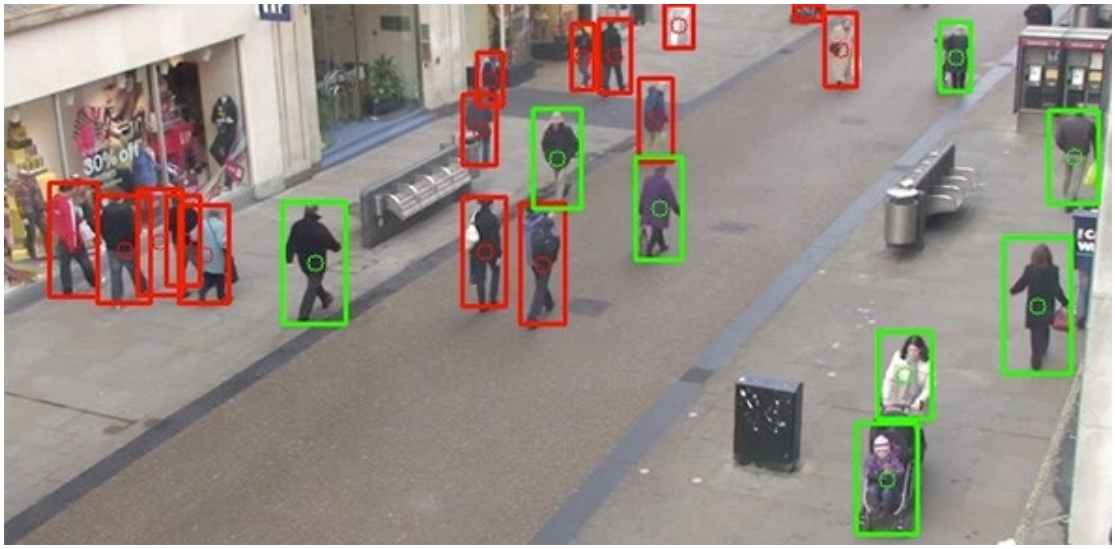
Video surveillance

Because state-of-the-art object detection techniques can accurately identify and track multiple instances of a given object in an exceedingly scene, these techniques naturally lend themselves to automating video surveillance systems.Object Detection algorithms have different applications in various fields such as defence, security, and health care

For instance, object detection models are capable of tracking multiple people at once without delay, in real-time, as they move through a given scene or across video frames. From retail stores to industrial factory floors, this type of granular tracking could provide invaluable insights into security, worker performance and safety, retail traffic, and more.

Images showing the video surveillance





Crowd Counting

Crowd counting is another valuable application of the object detection. For densely populated areas such as theme parks, malls, and city squares, object detection can help several businesses and municipalities more effectively to measure different varieties of traffic—whether on foot, in vehicles, or otherwise.

This ability to localize and track people as they maneuver through various spaces could help businesses optimize anything from the logistics pipelines and inventory management, to store hours, to shift scheduling, and more. Similarly, object detection could help cities to plan events, dedicate municipal resources, etc.

Self Driving Cars

Real-time car detection models are key to the success of autonomous vehicle systems. These systems must be able to identify, locate, and track objects around them so as to maneuver through the globe safely and efficiently.

And while tasks like image segmentation is (and often are) applied to autonomous vehicles, object detection remains a foundational task that underpins current work on making the self-driving cars a reality.

Real-time parking occupancy and availability

AI-based smart parking solutions include special IoT tools which can count the amount or the number of parked vehicles and empty parking spaces in a car parking zone. It detects if there's a car presence in in a parking space and then sends the information to the management platform. The best part of it is that this AI-based solution collects and formats the information in real-time.

Benefits for the parking managers

That real-time information about parking occupancy and availability are often observed or analyzed later. Every parking manager, parking analyst or decision-maker can optimize this parking management strategy at any time.

Benefits for the parking users

We all know that there are too many vehicles and there is not enough parking spaces.But thanks to the real-time data about parking occupancy and availability,parking users can easily find where to park their vehicle.That eliminates one in all the biggest parking problems – circling for an empty automobile parking space and city traffic.

Parking Telecom’s AI-based smart parking solutions

There is no doubt that the artificial intelligence has a big impact on parking management in our digital age. The AI-based parking management solutions will help for better parking experience, parking spaces control, convenience, flow speed and city traffic.

Parking Telecom’s AI-based parking solutions and systems are created to unravel or solve every parking management problem! We understand the difficulties that parking managers usually face and we know the way to resolve them.

11 METHODOLOGY IN VIDEO PROCESSING

The operation follows 5 steps namely: video acquisition, pre-processing, Thresholding, Morphological operation, car detection & region of interest calculation

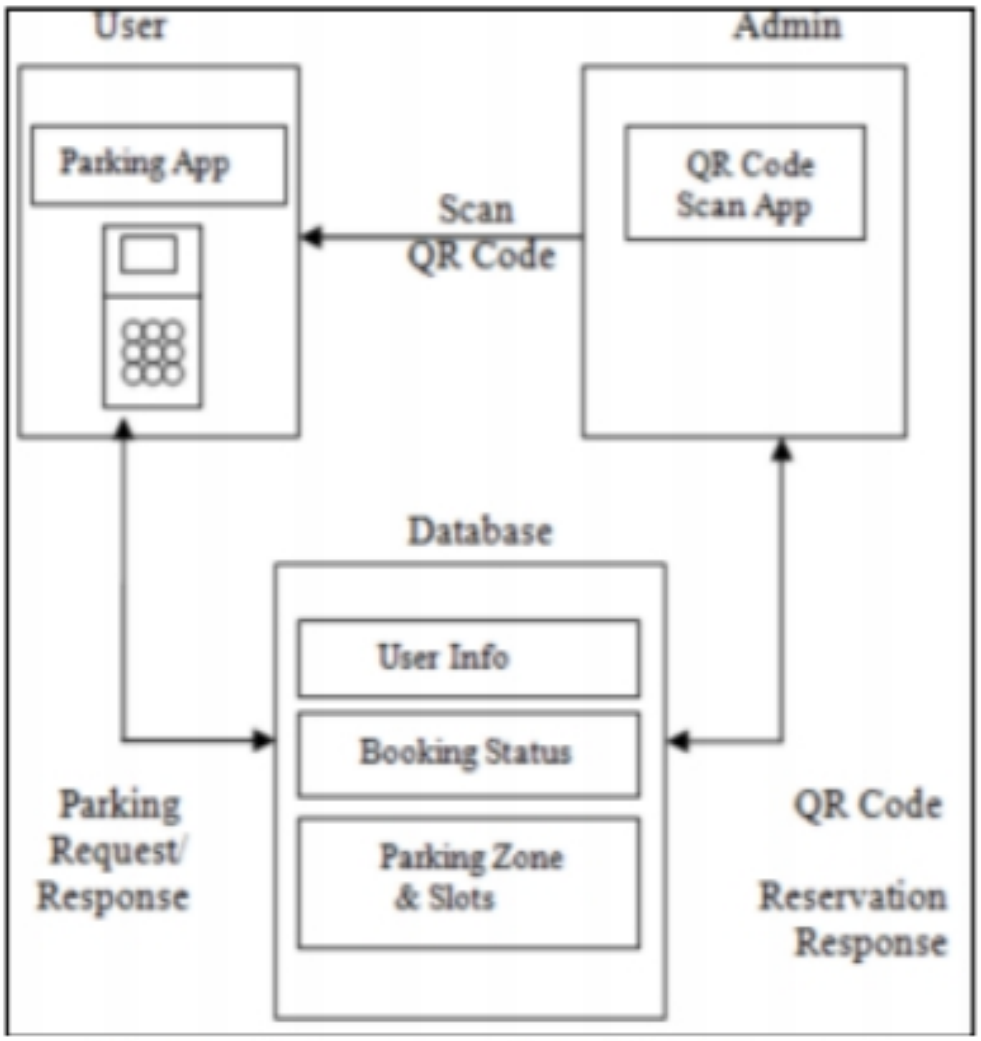


Figure 1: Layout of Parking System

Block Diagram

11.1 FLOW CHART.

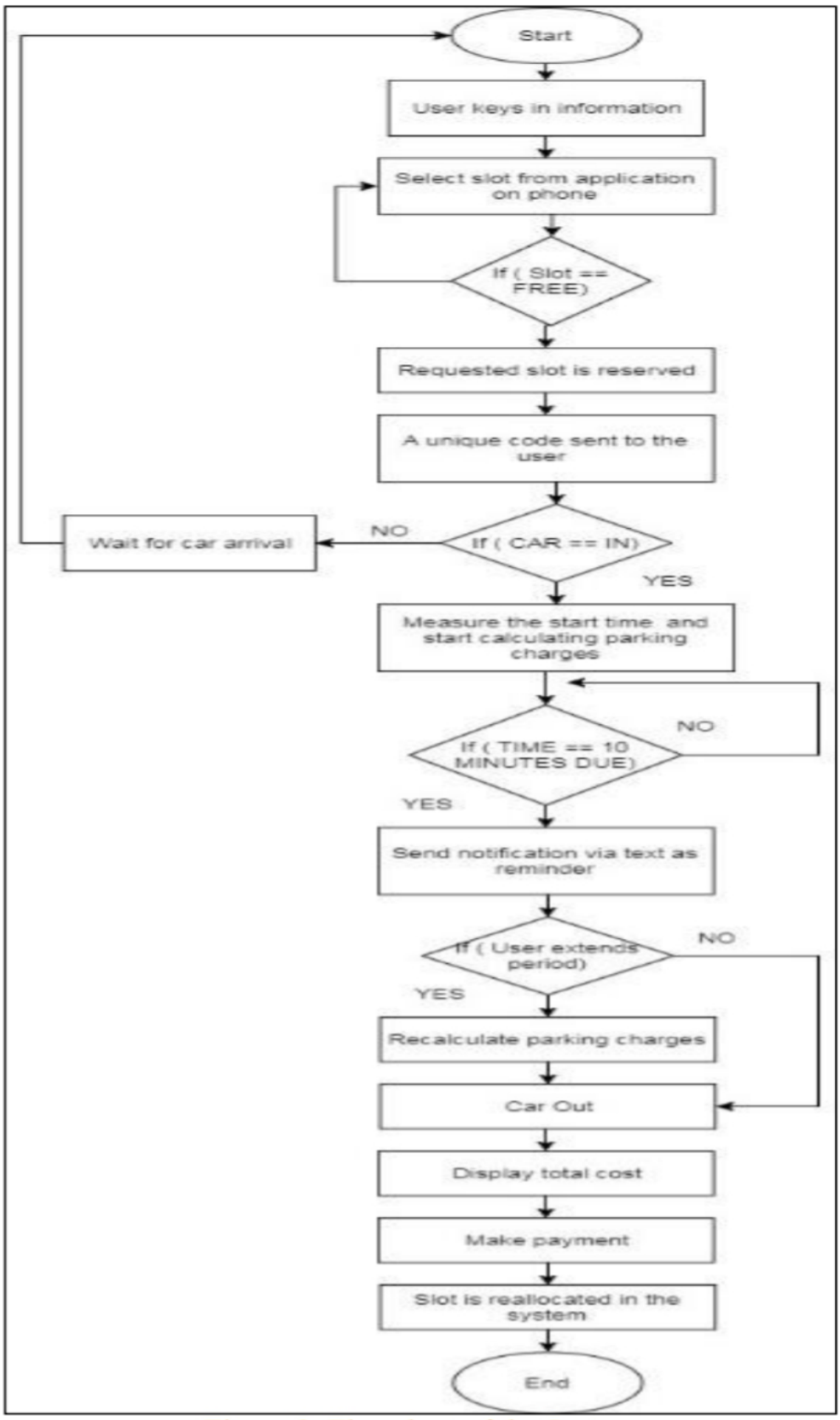


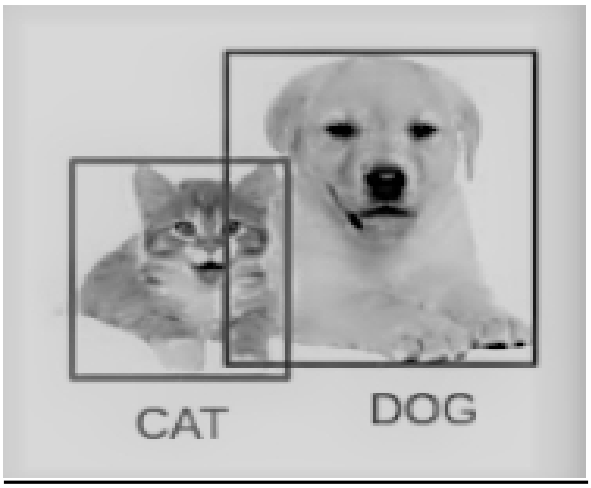
Figure 3: Flowchart of the System

11.3 Masked RCNN

RCNN stands for Region Based Convolutional Neural Networks. **Mask RCNN** is a deep neural network for solving instance segmentation problem in computer vision or machine learning . In other words, it has the ability to separate different types of objects in a video or a image. You give it a video or an image, it gives you the object bounding the boxes, classes and **masks**.
This mask localizes the most important aspects of each input for prediction of the network which is original.Masks are created by a secondary network whose goal is creating as small as possible an explanation while still preserving the accurate original network.
Mask R-CNN is used widely for detection of objects. For a given image, it returns the class label and coordinates of bounding box for each and every object in the image.

Input image:

Output image:



The Mask R-CNN framework is built on top of Faster R-CNN. So, in the given image, Mask R-CNN, in addition with the class label and the bounding box coordinates for each of the object, will return object mask too.Deep learning-based object detection models,they typically have two different parts. An encoder takes an image as an input and runs it through a series of layers and blocks that learn to extract statistical features and is used to locate and label the objects. Outputs from the encoder are then passed to a decoder, which then predicts bounding boxes and labels for each object.

Working of Faster R-CNN: This will help us grasp the intuition behind the Mask R-CNN as well.

1. The Faster R-CNN first uses a Convolutional Network to extract the feature maps from the provided images

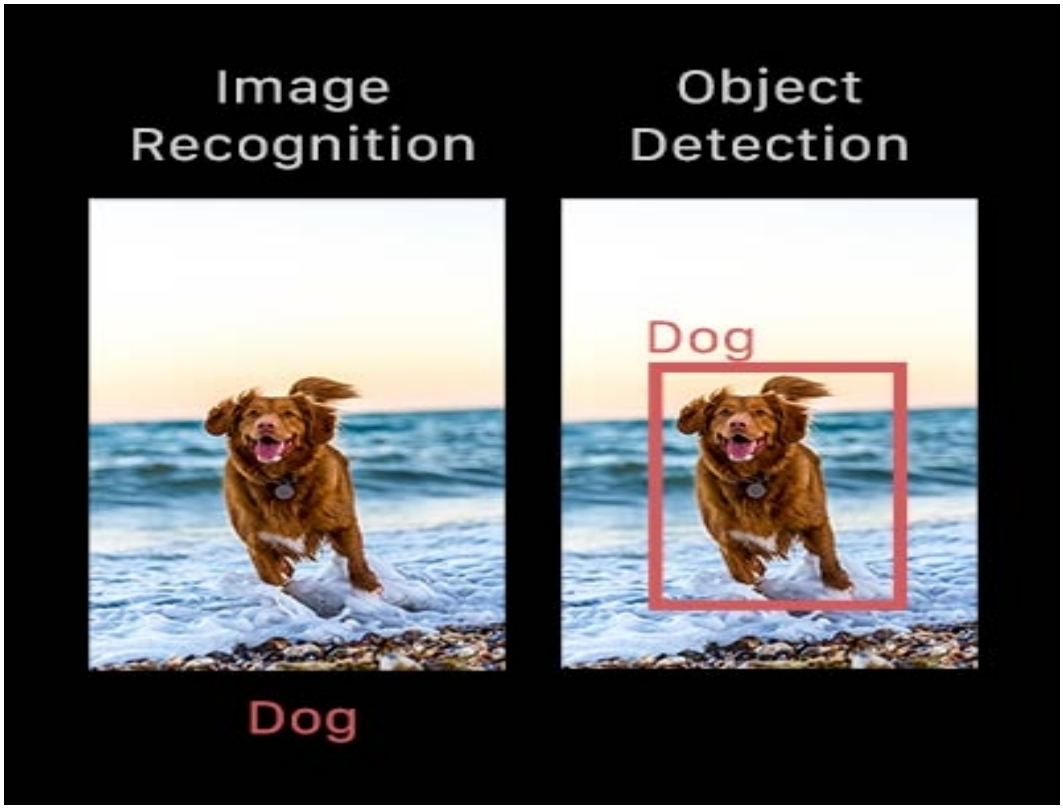
2. These provided feature maps are then passed through a Region of Proposal Network (RPN) which in turn returns the candidate bounding the boxes

3. Then we apply an RoI pooling layer on these given candidate bounding boxes to bring all the candidates to the same size

4. And then finally, the proposals are passed through a fully connected layer to classify and output the bounding boxes for the given objects

11.4 Backbone Model

Similar to the Convolutional Network which we use in Faster R-CNN to extract the feature maps from the image, we use the ResNet architecture to extract the features from the images in the Mask R-CNN. So, the first step taking an image and extracing features using the ResNet architecture. These features act as input for the following next layer.



11.5 Region Proposal Network (RPN)

Now, we have to take the feature maps which is obtained in the previous step and apply a region proposal network (RPM). This basically predicts if an given object is present in that particular region (or not). In this particular step, we get those regions or feature maps which as the model predicts contain some object.

11.6 Region of Interest (RoI)

The regions which is obtained from the RPN might be of different shapes, right? Hence, we apply a pooling layer and convert all the regions to the same shape. Next, these regions are passed through a fully connected layer or network so that the bounding boxes and the class label are predicted.

Untill this point, the steps are almost similar to how Faster R-CNN works.But Now comes the difference between the two frameworks.

Apart from this, **Mask R-CNN also generates the segmentation mask.**

For that, we first compute the region of interest so that the computational time can be reduced. For all the predicted regions, we have to compute the Intersection over Union (IoU) with the ground truth boxes. We can computer IoU by:

IoU = Area of the intersection / Area of the union

Now, only if the IoU is greater than or equal to 0.5, we consider that as a region of interest. Orelse, we neglect that particular region. We do this for all the given regions and then select only a set of regions for which the IoU is greater than 0.5.

11.7 Segmentation Mask

We have the RoIes based on the IoU values, now we can add a mask branch to the already existing architecture. Now this returns the segmentation mask for each of the region that contains an object. It returns a mask of size 28 X 28 for each of the region which is then scaled up for inference.

11.8 Crowd counting

Crowd counting is considered another important application of object detection. For densely populated areas like parks, malls,stations and city squares, the object detection helps businesses and municipalities more effectively to measure different kinds of traffic—whether in vehicles, or otherwise.

This ability to track people as they move through various spaces could benifit in businesses and optimize anything from inventory management and logistics pipelines, to store hours, to shift scheduling, and etc. Similarly, object detection could help in cities plan events, dedicate municipal resources, etc

11.9. Steps to implement Mask R-CNN

We will be using the mask RCNN framework created by the Data scientists .Now the steps by which we will perform image segmentation using Mask R-CNN.

Step 1: Repository cloning

First, we will have to clone the mask RCNN repository which contains the architecture required for the Mask R-CNN. Once this is done, we will install the dependencies required by Mask R-CNN.

Step 2: Installing the dependencies

Here is a list of all the dependencies for Mask R-CNN:

- *numpy*
- *twillow*
- *matplotlib*
- *scikit-image*
- *tensorflow*
- *keras*
- *opencv-python*
- *IPython*

You must install all these dependencies before using the Mask R-CNN framework.

Step 3: Download the pre-trained weights

Next, we have to download the previously trained weights. These weights can be obtained from a model that is trained from the MS COCO dataset. Once you have downloaded the weights, we have to paste this file in the samples folder of the Mask_RCNN repository that we cloned in step 1.

Step 4: Prediction of our image

In the last step, we will have to use the Mask R-CNN architecture and the previously trained weights to generate all predictions for our own images.

Once we're done with all these four steps, it's time to go to the Jupyter Notebook and we will implement all these things in Python and then we have to generate the masks along with the classes and bounding the boxes for objects in our project images.

12 .smart parking system spot detection

In this project we had to detect whether the parking spot is available or occupied based on security view camera photos. There are limitations to our work, but once these are resolved this project will be a low-cost solution for optimizing parking availability. There's certainly a potential in this project to simplify the process of finding a parking spot in a given region since because most of the parking lots have many security cameras installed and this way the parking lots won't need to have any additional equipment needed to be installed.

Parking Lot dataset is available on Kaggle which has enough data points for training a deep learning model and all xml files with annotations whether a particular spot has been occupied or not. You can access the dataset of parking lot dataset. For the model, we used state-of-the-art object detection and segmentation Mask-RCNN model which performs greatly.

As we can see the Mask-RCNN pre-trained on COCO dataset model does an extraordinarily excellent job at image object detection and segmentation. Although in some cases it misclassifies .We will create a class Parking Lot that will reload the dataset, and extract the contours of the bounding boxes by the parsing of annotated files, and create the masks based on the extracted contours and then we'll need an image to reference function that will return the path to the annotated file.

We can train these model from begining using the whole set of data and see how it performs, and then we can try tuning detection

threshold. But ideally, more varying dataset is essential with complete annotation is fundamental to create an accurate parking spot detection model. Furthermore, if we want to build not only accurate but also robust detection model, it is really important to account for cars that are moving and not yet parked yet, unauthorized parked cars. These maybe further steps to look into in the future.

13.CODE SNIPPET

```
## mounting google drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[ ] import os
    os.chdir('/content/drive/My Drive/MASKED_RCNN')
```

```
[ ] !ls
```

```
AI_Car_parking_video.mp4  Mask_RCNN
```

```
[ ] ## since we are using Mask RCNN it is build sarrounded by keras 2.0.8 and tensorflow 1.3 so we need to update the keras and tensorflow
    !pip install keras==2.0.8
```

```
Collecting keras==2.0.8
  Downloading https://files.pythonhosted.org/packages/67/3f/d117d6e48b19fb9589369f4bdbe883aa88943f8bb4a850559ea5c546fefb/Keras-2.0.8-py2.py3-none-any.whl (276kB)
     |████████████████████████████████| 276kB 8.4MB/s
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.7/dist-packages (from keras==2.0.8) (1.4.1)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.7/dist-packages (from keras==2.0.8) (3.13)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.7/dist-packages (from keras==2.0.8) (1.19.5)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from keras==2.0.8) (1.15.0)
ERROR: textgenrnn 1.4.1 has requirement keras>=2.1.5, but you'll have keras 2.0.8 which is incompatible.
Installing collected packages: keras
  Found existing installation: Keras 2.4.3
    Uninstalling Keras-2.4.3:
      Successfully uninstalled Keras-2.4.3
```

```
[ ] !pip install twilio
```

```
Collecting twilio
  Downloading https://files.pythonhosted.org/packages/c7/c2/aad9b42e2513a09b10630558a392d557d193feeece1b1d0bb3154b4d4c13/twilio-6.53.0.tar.gz (470kB)
     |████████████████████████████████| 471kB 7.2MB/s
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from twilio) (1.15.0)
Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages (from twilio) (2018.9)
Collecting PyJWT==1.7.1
  Downloading https://files.pythonhosted.org/packages/87/8b/6a9f14b5f781697e51259d81657e6048fd31a113229cf346880bb7545565/PyJWT-1.7.1-py2.py3-none-any.whl
Requirement already satisfied: requests>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from twilio) (2.23.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2.0.0->twilio) (2020.12.5)
Building wheels for collected packages: twilio
  Building wheel for twilio (setup.py) ... done
  Created wheel for twilio: filename=twilio-6.53.0-py2.py3-none-any.whl size=1251197 sha256=2390e24fbf1fbe3a9e55bf8a3d6c5467c80e50a9547764d6ba08cade38de5a03
  Stored in directory: /root/.cache/pip/wheels/80/92/b0/f7fcc5d56edd9d23fe35c73400c6f600aefe736e05a9e70806
Successfully built twilio
Installing collected packages: PyJWT, twilio
Successfully installed PyJWT-1.7.1 twilio-6.53.0
```

```
[ ] import sys
    import os

    root_dir=os.path.abspath('/content/drive/MyDrive/MASKED_RCNN/Mask_RCNN')
    sys.path.append(root_dir)
    sys.path
```

```
[ ] ## installing requirement txt file
    !pip install -r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements.txt
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from -r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements.txt (line 1)) (1.19.5
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from -r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements.txt (line 2)) (1.4.1)
Requirement already satisfied: Pillow in /usr/local/lib/python3.7/dist-packages (from -r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements.txt (line 3)) (7.0.0
Requirement already satisfied: cython in /usr/local/lib/python3.7/dist-packages (from -r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements.txt (line 4)) (0.29.
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from -r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements.txt (line 5)) (3
Requirement already satisfied: scikit-image in /usr/local/lib/python3.7/dist-packages (from -r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements.txt (line 6))
Requirement already satisfied: tensorflow>=1.3.0 in /usr/local/lib/python3.7/dist-packages (from -r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements.txt (line
Requirement already satisfied: keras>=2.0.8 in /usr/local/lib/python3.7/dist-packages (from -r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements.txt (line 8))
Requirement already satisfied: opencv-python in /usr/local/lib/python3.7/dist-packages (from -r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements.txt (line 9))
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages (from -r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements.txt (line 10)) (2.10.0
Requirement already satisfied: imgaug in /usr/local/lib/python3.7/dist-packages (from -r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements.txt (line 11)) (0.2.
Requirement already satisfied: IPython[all] in /usr/local/lib/python3.7/dist-packages (from -r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements.txt (line 12))
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->-r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requiremer
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->-r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements.tx
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->-r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/require
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->-r /content/drive/MyDrive/MASKED_RC
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image->-r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirement
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image->-r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirem
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image->-r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirements
Requirement already satisfied: astor>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow>=1.3.0->-r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requirem
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow>=1.3.0->-r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requireme
Requirement already satisfied: tensorboard<1.14.0,>=1.13.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow>=1.3.0->-r /content/drive/MyDrive/MASKED_RCNN/Mas
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow>=1.3.0->-r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requ
Requirement already satisfied: tensorflow-estimator<1.14.0rc0,>=1.13.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow>=1.3.0->-r /content/drive/MyDrive/MAS
Requirement already satisfied: absl-py>=0.1.6 in /usr/local/lib/python3.7/dist-packages (from tensorflow>=1.3.0->-r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requir
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.7/dist-packages (from tensorflow>=1.3.0->-r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requireme
Requirement already satisfied: protobuf>=3.6.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow>=1.3.0->-r /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/requi
Requirement already satisfied: keras-applications>=1.0.6 in /usr/local/lib/python3.7/dist-packages (from tensorflow>=1.3.0->-r /content/drive/MyDrive/MASKED_RCNN/Mask_
```

```
[ ] !pip install opencv-python
```

```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.7/dist-packages (4.1.2.30)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (from opencv-python) (1.19.5)
```

```
[ ] import numpy as np
    from numpy.random import randn
    import matplotlib.pyplot as plt
    from matplotlib import rcParams
    import cv2
    import keras
    import tensorflow
    import seaborn as sns
```

```
[ ] import os
    os.chdir("/content/drive/My Drive/MASKED_RCNN/Mask_RCNN/")
```

```
[ ] import mrcnn.config
    import mrcnn.utils
    from mrcnn.model import MaskRCNN
    from pathlib import Path
    from twilio.rest import Client
```

```
[ ] ## GPU confugaration ## maskRCNN configaration
    class interferenceconfig(mrcnn.config.Config):
      NAME = "coco_pretrained_model_config"
      IMAGES_PER_GPU = 1
```

```
NAME = "coco_pretrained_model_config"
IMAGES_PER_GPU = 1
GPU_COUNT = 1
NUM_CLASSES = 1 + 80
DETECTION_MIN_CONFIDENCE = 0.6

config= interferenceconfig()
config.display()
```

```
Configurations:
BACKBONE                   resnet101
BACKBONE_STRIDES           [4, 8, 16, 32, 64]
BATCH_SIZE                 1
BBOX_STD_DEV               [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE     None
DETECTION_MAX_INSTANCES    100
DETECTION_MIN_CONFIDENCE   0.6
DETECTION_NMS_THRESHOLD    0.3
FPN_CLASSIF_FC_LAYERS_SIZE 1024
GPU_COUNT                  1
GRADIENT_CLIP_NORM         5.0
IMAGES_PER_GPU             1
IMAGE_CHANNEL_COUNT        3
IMAGE_MAX_DIM              1024
IMAGE_META_SIZE            93
IMAGE_MIN_DIM              800
IMAGE_MIN_SCALE            0
IMAGE_RESIZE_MODE          square
IMAGE_SHAPE                [1024 1024    3]
LEARNING_MOMENTUM          0.9
LEARNING_RATE              0.001
```

```
## making bounding box of the car
def get_car_boxes(boxes,class_ids): ## class_ids is the labels of the MaskRCNN
  car_boxes=[]
  for i,box in enumerate(boxes): ## each box of the free space we have to detect what is present in that free space
    if class_ids[i] in [3,8,6]: ## 3,8,6 are the label whereas 3 indicate the car,8 indicate the truck and 6 may be the person
      car_boxes.append(box)
  return np.array(car_boxes)
```

```
## twilio config
twilio_acc_sid="ACf5b2a892762d5c97bdb07583dd13a225"
twilio_auth_token="59eab84f49d3f00e6594479253b62147"
twilio_ph_no="+8283736953"
twilio_destination_no="+8617864426"
client=Client(twilio_acc_sid,twilio_auth_token)
```

```
## importing coco datasets
sys.path.append(os.path.join(root_dir,"/content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/samples/coco"))
import coco
model_dir=os.path.join(root_dir, "logs")
coco_model_path= os.path.join(root_dir,"mask_rcnn_coco.h5")
```

```
if not os.path.exists(coco_model_path):
  mrcnn.utils.download_trained_weights(coco_model_path)
```

```
Downloading pretrained model to /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/mask_rcnn_coco.h5 ...
... done downloading pretrained model!
```

```
[ ] if not os.path.exists(coco_model_path):
        mrcnn.utils.download_trained_weights(coco_model_path)
```

```
Downloading pretrained model to /content/drive/MyDrive/MASKED_RCNN/Mask_RCNN/mask_rcnn_coco.h5 ...
... done downloading pretrained model!
```

```
[ ] model=MaskRCNN(mode='inference',model_dir=model_dir,config=config)
    model.load_weights(coco_model_path,by_name=True)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/tensorflow_backend.py:1154: calling reduce_max_v1 (from tensorflow.python.ops.math_ops) with
Instructions for updating:
keep_dims is deprecated, use keepdims instead
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/tensorflow_backend.py:1188: calling reduce_sum_v1 (from tensorflow.python.ops.math_ops) with
Instructions for updating:
keep_dims is deprecated, use keepdims instead
WARNING:tensorflow:From /content/drive/My Drive/MASKED_RCNN/Mask_RCNN/mrcnn/model.py:772: to_float (from tensorflow.python.ops.math_ops) is deprecated and will be remove
Instructions for updating:
Use tf.cast instead.
```

```
[ ] video_source="/content/drive/MyDrive/MASKED_RCNN/AI_Car_parking_video.mp4"

    ## spotting park spaces
    parked_car_box=None
    car_box=None
```

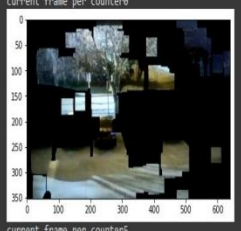
```
[ ] video_cap=cv2.VideoCapture(video_source)
    free_space_frame=0
    sms_sent=False
    count=0
    temp=np.array(4,)
    parked_car_box1=[None]*11

    def checkEqual2(iterator):
      print(iterator)


    ## loop over each frame in the video
    counter = 0
    ref_image = None
    ref_image_color = None
    cumm_image = None
    img_thresh_sub1 = None
    final_ref = None

    while True:
      ret,frame=video_cap.read()

      if not ret:
        print("couldn't read video")
        break

      elif counter<40:
        ## let's say the video having fps of 30 that means frame per second at much speed images is shown. Here for 40 frames we are checking at 1 sec a
        ## and then 1.1 second we are checking possibility of motion by MaskRCNN by creating two different video object two find out the difference
        ## in background pixel to pixel.So that we can able to detect the movement of the pixel.
        ## creating another video object to compare the two different frame to find out the possibility of motion
        ret,frame1=video_cap.read()
        d=cv2.absdiff(frame,frame1)
        grey=cv2.cvtColor(d,cv2.COLOR_BGR2GRAY)
        blur=cv2.GaussianBlur(grey,(1,1),0)
        ret,th=cv2.threshold(blur,20,255,cv2.THRESH_BINARY)

        ## performing the morphological operation
        dilated=cv2.dilate(th,np.ones((30,30),np.uint8),iterations=1)
        erode=cv2.erode(dilated,np.ones((30,30),np.uint8),iterations=1)
```

```
    erode=cv2.erode(dilated,np.ones((30,30),np.uint8),iterations=1)

    ## fill the contours for better morphing of the vechile
    c,_=cv2.findContours(erode,cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    frame1=cv2.drawContours(frame1, c, -1, (0,0,0), cv2.FILLED)

    ## now we will checking and visualize every five frame
    if counter%5==0:
      print("current frame per counter"+str(counter))
      plt.imshow(frame1)
      plt.show()

    counter=counter+1 ## here that means we will continue for 40fps for checking the possibility of motion and increase the count by 1 after this
    continue      ## execution will be stop here.

  ## converting BGR colour to RGB color
  if counter==40:
    rgb_img=frame1[:,:,::-1]
    counter+=1
  else:
    rgb_img=frame[:,:,::-1]

  result=model.detect([rgb_img],verbose=0)
  r=result[0]

  if parked_car_box is None:
    print("going to mark vechile frame number:",counter)
    video_cap=cv2.VideoCapture(video_source)
    ## this is the first frame assume all the car detected in parking spaces and save the location of each car in parking space box and go to the
    ## next frame
    parked_car_box=get_car_boxes(r['rois'],r['class_ids'])
  else:
    ## get where the car is currently located in the frame
    car_box=get_car_boxes(r['rois'],r['class_ids'])

    ## see how much car overlaps with the known parking space

    overlaps= mrcnn.utils.compute_overlaps(parked_car_box, car_box) ## sometimes in parking area the cars are veru close to each other so while
    ## detecting car boxes may overlap with each other . So by MaskRCNN we detect that overlaps.

    ## assume no spaces are free until we find out the new one
```

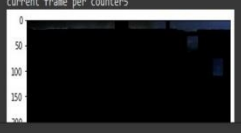
```
    ## assume no spaces are free until we find out the new one
    free_space=False

    ## now to clearly visualize the overlaps
    for parking_area,overlap_area in zip(parked_car_box,overlaps):
      ## find out the max overlap
      max_overlap=np.max(overlap_area)

      ## finding out of the cordinates of left bottom,right bottom,right and left upper of the parking area
      y1,x1,y2,x2=parking_area

      ## if the max_overlaps are less than particular threshold value that means parking space is available and making with green colour
      if max_overlap<0.15:
        cv2.rectangle(frame,(x1,y1),(x2,y2),(0,255,0),3)
        free_space=True

      else:
        cv2.rectangle(frame,(x1,y1),(x2,y2),(0,0,255),1) ## parking space is still occupied and marking by red box

      ## write the max overlaps measurement inside the box
      font=cv2.FONT_HERSHEY_DUPLEX
      cv2.putText(frame, f"{max_overlap:0.2}", (x1 + 6, y2 - 6), font, 0.3, (255, 255, 255))

      ## checking the parking space
      if free_space:
        free_space_frame += 1

      else:
        free_space_frame += 0 ## else reset to zero

      ## if the space is free for the several frames then it is really free
      if free_space_frame>120:
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, f"SPACE AVAILABLE!", (10, 150), font, 3.0, (0, 255, 0), 2, cv2.FILLED)

        ## sending sms
        if not sms_sent:
          print("SENDING SMS!!!")
          message=client.messages.create(body='PARKING SPACE AVAILABLE!!!',from_=twilio_ph_no,to=twilio_destination_no)
          sms_sent=True
          print("Hope to get a message to your respective phone number")
```

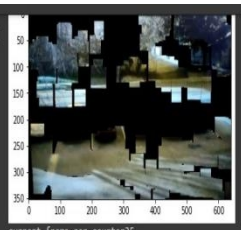
```
[ ]        ## sending sms
           if not sms_sent:
             print("SENDING SMS!!!")
             message=client.messages.create(body='PARKING SPACE AVAILABLE!!!',from_=twilio_ph_no,to=twilio_destination_no)
             sms_sent=True
             print("Hope to get a message to your respective phone number")


       ## showing the video frame and saving each frame
         name = str(count) + ".jpg"
         name = os.path.join('./ak', name)
         cv2.imwrite(name, frame)
         count+=1

         #'q' to quit
         if cv2.waitKey(1) & 0xFF == ord('q'):
           break


    print("Video finished")
    video_cap.release()
```

current frame per counter0

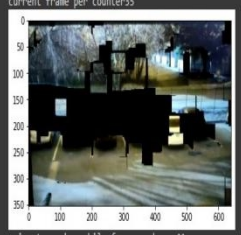


current frame per counter5





current frame per counter35



going to mark vechile frame number: 41
SENDING SMS!!!

```
[ ] ## creating a video including all the frame and the whole demonstration of the project in a single video
    import glob
    import os
    import sys
    images=list(glob.iglob(os.path.join('./MASKED_RCNN', '*.*')))
    images=sorted(images, key=lambda x: float(os.path.split(x)[1][:-3]))

    ## outvid = output video,images =list of images use in the video, fps= frame per second,size=size of
                ## each frame

    def make_video(outvid, images=None, fps=30, size=None,is_color=True, format="FMP4"):
      from cv2 import VideoWriter,VideoWriter_fourcc,imread,resize
      fourcc=VideoWriter_fourcc(*format)
      vid=None
      for image in images:
```

```
[ ] ## creating a video including all the frame and the whole demonstration of the project in a single video
    import glob
    import os
    import sys
    images=list(glob.iglob(os.path.join('./MASKED_RCNN', '*.*')))
    images=sorted(images, key=lambda x: float(os.path.split(x)[1][:-3]))

    ## outvid = output video,images =list of images use in the video, fps= frame per second,size=size of
                ## each frame

    def make_video(outvid, images=None, fps=30, size=None,is_color=True, format="FMP4"):
      from cv2 import VideoWriter,VideoWriter_fourcc,imread,resize
      fourcc=VideoWriter_fourcc(*format)
      vid=None
      for image in images:
        if not os.path.exists(image):
          raise FileNotFoundError(image)
        img=imread(image)
        if vid is None:
          if size is None:
            size=img.shape[1],img.shape[0]
          vid=VideoWriter(outvid,fourcc,float(fps),size,is_color)

        if size[0]!=img.shape[1] and size[1]!=img.shape[0]:
          img=resize(img,size)
        vid.write(img)

      vid.release()
      return vid

    make_video(' ',images,fps=30)
```

14 .Future Scope

Object detection has great scope in the wide area:-

1) It is used by lots of industries specially in automotive industry in self driving car for automatic pedestrian detection

2) Amid pandemic situation it has lots of scope for corona virus detection

3) It can be used for video surveillance .

4) It has lots of scope for face recognition or face unlock in car.

5) By this methodology we can use this application for geospatial purpose.

15 .CONCLUSION:

So finally after reading the car parking video and analysis of each and every frame of the video at particular fps by using computer vision technique and by using of advanced Masked RCNN technique we can easily identify whether the parking space available or not in the particular parking lot by calculating region of interest of the parking space and car detection .Hence finally we can easily identify the parking space and that relevant information has successfully sent to the user. Hence the car driver can easily able to park his car in the particular parking space without any hindrance .

16. APPENDIX

The team comprised of five members and each one of them had contributed well on their part.The efficiency of the team is surely reflected in the work showcased above.

ANKIT SARKAR (1707178,EEE-3)-The mind behind the idea of shifting the project towards smart parking system in future and get a device out of it. Had equal contribution in study of the IEEE papers.Took the initiative to design the presentation for the same .Worked out with the code to implement video processing.

SAIKAT CHAKRABORTY (1707218,EEE-3) -had some role in each and every step of the project as it progressed with time. Had contibution in implementation of the Masked RCNN model and the code implementation of opencv and also further contribution of image processing and analysis designing and planning.Took the initiative to design the presentation for the same with the help of the others member .

SOUVIK MAL (1707228,EEE-3)- had some role in each and every step of the project as it progressed with time. Had contribution in writing documents and ppt presentation planning.Took initiative in code analysis of the Video processing techniques and architecture designing or the same with the help of the others member .

SUMAN MONDAL (1707237,EEE-3)-had some role in each and every step of the project as it progressed with time.Had contribution for studying and construct of the renowned research papers and get the idea as to how to implement the project in terms of documentation , writing the research paper. . Project planning and methodology lying under the work exhibited.

ABHISEK PATI(1707251, EEE-3) - had some role in each and every step of the project as it progressed with time. The start included study and construe of the renowned research papers and get the idea as to how to implement the project in terms of documentation , writing the research paper.Analysis of the research papers has been done and their outcomes are compared . Project planning and methodology lying under the work exhibited.

17. References :

1: https://www.ideals.illinois.edu/handle/2142/50380

2: https://www.fritz.ai/object-detection/#:~:text=Object%20detection%20is%20a%20computer,all%20while%20accurately%20labeling%20them

3: https://www.slideshare.net/AmitShukla100/smart-car-parking-system-93802531

FIG 4: https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/