

***TITLE: AI ENABLED CAR
PARKING USING OpenCV***

REPORT FORMAT

1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

2. LITERATURE SURVEY

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map

3.2 Ideation & Brainstorming

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

4.2 Non-Functional requirements

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

5.2 Solution Architecture

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

6.2 Sprint Planning & Estimation

7. CODING & SOLUTIONING

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema (if Applicable)

8. PERFORMANCE TESTING

8.1 Performance Metrics

9. RESULTS

9.1 Output Screenshots

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

Project Overview:

Introduction:

Crafted as an advanced parking facilitator, the AI-Driven Parking System stands as a cutting-edge innovation aimed at optimizing the vehicle parking process. Leveraging Artificial Intelligence (AI) and Computer Vision methodologies, this solution redefines the way vehicles are parked, ensuring a seamless and efficient parking experience.

Objective:

At the forefront of our mission lies the development of an intelligent parking system that capitalizes on OpenCV, a widely acclaimed computer vision library. Our primary goal is to achieve precise vehicle detection and autonomous guidance into parking spaces, establishing a smart parking infrastructure that ensures accuracy and convenience for drivers.

Key Features:

1. Live vehicle detection and continuous tracking employing OpenCV's capabilities.
2. Automated guidance system facilitating seamless parking for drivers.
3. Seamless integration with an intuitive interface catering to both drivers and parking operators.
4. Comprehensive data logging and reporting functionalities enabling meticulous monitoring and in-depth analysis of parking metrics.
5. Real-time occupancy monitoring and management, allowing for efficient utilization of parking spaces.
6. Adaptive pricing algorithms based on demand and occupancy levels, optimizing revenue generation for parking operators.

Technologies and Tools:

1. OpenCV: Harnessing the power of the OpenCV library for proficient image processing, precise object detection, and continuous tracking.
2. Python: The primary programming language employed for system implementation.
3. GUI Framework (optional): Potential integration for crafting a user-centric interface.
4. Hardware (optional): Cameras facilitating the capture of real-time footage for enhanced system functionality.

Components:

- a. Vehicle Detection and Tracking: Implementing advanced object detection algorithms for real-time identification of vehicles within the camera feed. Employing tracking algorithms to monitor and trace vehicle movements across the parking area.
- b. Parking Guidance System: Deploying AI-driven algorithms to compute optimal parking spots based on vehicle dimensions and available space. Offering drivers visual and/or auditory cues for seamless parking maneuvers.
- c. User Interface: Designing an intuitive and user-friendly interface (optional) to facilitate driver-system interaction. Furnishing essential details regarding parking space availability, guidance instructions, and feedback mechanisms.
- d. Data Logging and Reporting: Establishing a comprehensive system for logging parking statistics encompassing occupancy, duration, and frequency. Generating detailed reports to facilitate analysis and optimization strategies.

Workflow:

- a. **Vehicle Detection:** Acquire real-time video feeds via strategically positioned cameras within the parking area. Employ sophisticated image processing techniques to swiftly detect vehicles.
- b. **Tracking:** Implement robust tracking algorithms to monitor and trace the movements of identified vehicles.
- c. **Parking Guidance:** Analyze the availability of parking spaces based on detected vehicles and vacant areas. Offer drivers guidance through visual cues or audio prompts for efficient parking maneuvers.
- d. **User Interface (optional):** Develop an intuitive interface for seamless interaction between drivers and the system, enhancing user experience.
- e. **Data Logging and Reporting:** Systematically record parking data for comprehensive analysis and reporting purposes, facilitating strategic optimizations and improvements.

Testing and Validation:

Undertake comprehensive testing procedures aimed at validating the accuracy of vehicle detection and parking guidance functionalities.

Assess system performance across diverse conditions, including variations in lighting, weather, and vehicle sizes, ensuring robust performance and reliability across varied scenarios.

Benefits:

- 1. **Enhanced Parking Efficiency:** Streamlining parking processes, thereby reducing congestion and diminishing the time spent searching for available spaces, optimizing overall parking efficiency.
- 2. **Elevated User Experience:** Providing drivers with an enhanced experience, leading to heightened customer satisfaction through smoother parking procedures and improved convenience.

Purpose:

The aim of developing an AI-enabled car parking system utilizing OpenCV is to establish a cutting-edge parking infrastructure empowered by artificial intelligence and computer vision. The project is designed with specific objectives in mind:

1. **Efficient Space Utilization:** Leveraging computer vision techniques to accurately detect and track vehicles in real-time, optimizing parking space allocation for enhanced utilization of available spots.
2. **Improved User Experience:** Creating a driver-centric interface that offers guidance on parking availability, simplifying the parking process, and diminishing the time and frustration associated with finding suitable parking spaces.
3. **Reduced Congestion and Traffic:** Streamlining parking procedures to expedite and optimize parking contributes to reducing congestion in parking lots and adjacent roads, consequently enhancing traffic flow in busy urban settings.
4. **Enhanced Safety Measures:** By offering visual or auditory guidance to drivers, the project contributes to bolstering safety within parking areas. This guidance minimizes the risk of accidents or collisions during parking maneuvers, thus enhancing overall safety standards.

Ideation Phase

Brainstorm & Idea Prioritization Template

Brainstorm & Idea Prioritization Template:

Brainstorming provides a free and open environment that encourages to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome to develop a rich number of creative solutions.

Step -1 : Select the Problem Statement

Problem Statement:

The existing car parking systems in urban environments face significant challenges in terms of space optimization, time efficiency, and user convenience. Conventional systems often allocate parking spaces without considering vehicle dimensions, leading to inefficient use of available parking real estate. Drivers spend considerable time searching for suitable spots, causing congestion and increased environmental impact due to prolonged idling.

Maneuvering vehicles in confined spaces also poses safety concerns. Moreover, the lack of clear guidance exacerbates the parking process. Additionally, the absence of comprehensive data collection and analysis capabilities hinders parking operators' ability to make informed decisions about space allocation and utilization. In light of these issues, this project aims to develop an AI-enabled car parking system using OpenCV, employing computer vision and artificial intelligence techniques to address these challenges and enhance the overall parking experience for both drivers and operators.

The system will provide real-time vehicle detection and tracking, automated parking guidance, a user-friendly interface, and robust data logging and reporting functionalities

Step-2: Brainstorm, Idea Listing

Ideas requiring Manual Assistance

Crowdsourcing: Enable drivers to share real-time information about available parking spots through a dedicated app, fostering a collaborative platform for parking spot updates among users.

Parking Attendant Support: Employ attendants to guide and assist drivers in finding available spots.

Parking Reservation System: Introduce a system enabling drivers to pre-book parking spots in advance, streamlining the parking process and ensuring availability upon arrival.

Ideas requiring tools such as sensors

Ground-Level Sensors: Utilize ultrasonic or magnetic sensors at parking spots to detect vehicle presence, providing real-time updates on spot availability.

Parking Sensors with Lights: Install sensors triggering nearby lights to indicate vacant parking spots, offering clear visual cues for drivers seeking available spaces.

Ideas making using of AI and ML techniques

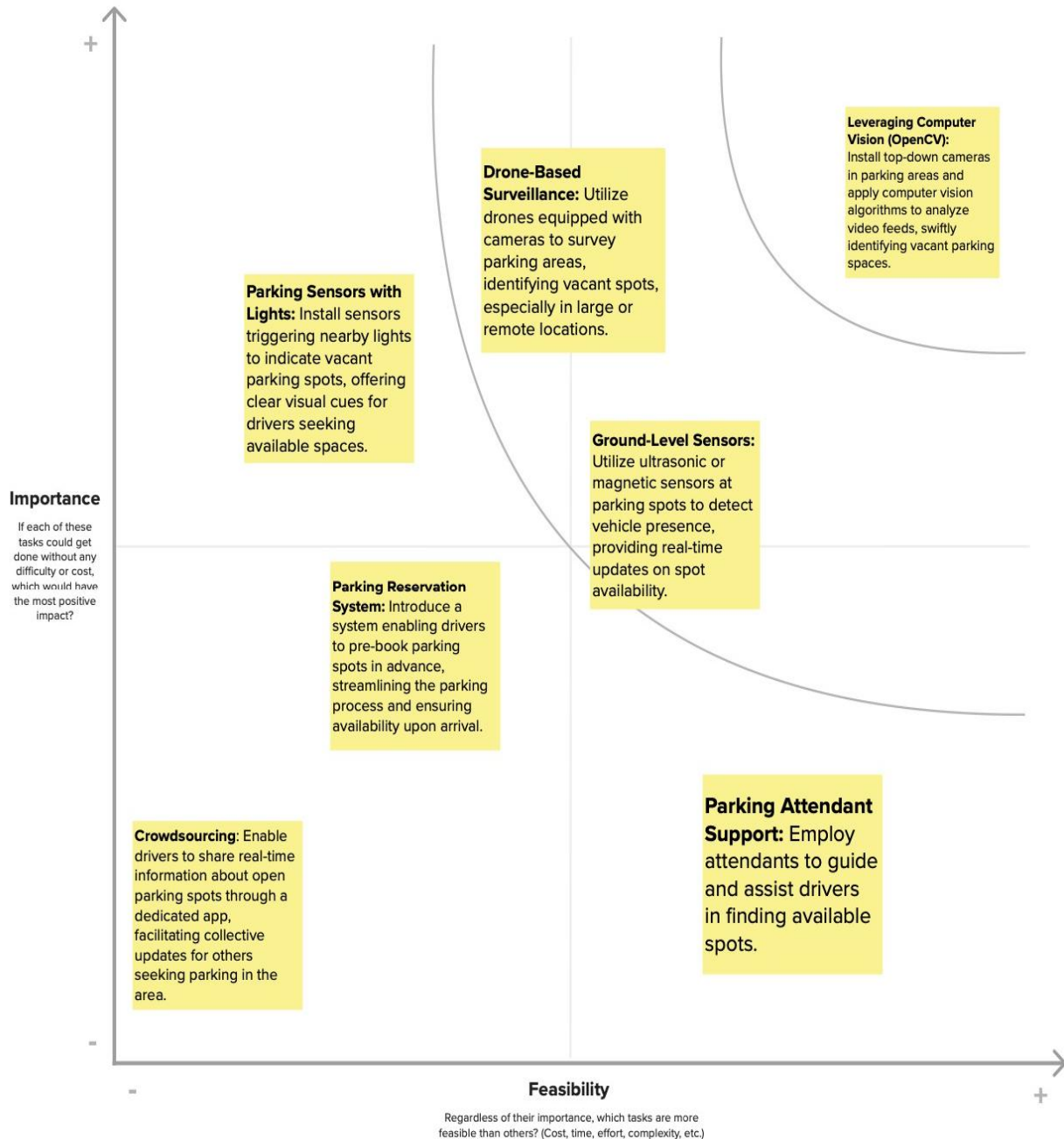
Parking Data Analytics: Analyze parking data to identify usage patterns and occupancy enabling data-driven parking management decisions.

Drone-Based Surveillance: Utilize drones equipped with cameras to survey parking areas, identifying vacant spots, especially in large or remote locations.

Leveraging Computer Vision (OpenCV): Install top-down cameras in parking areas and apply computer vision algorithms to analyze video feeds, swiftly identifying vacant parking spaces.

Parking Spot Markings: Implement highly visible and distinct markings for parking spots, simplifying the identification of vacant spaces for drivers.

Step-3: Idea Prioritization



Ideation Phase

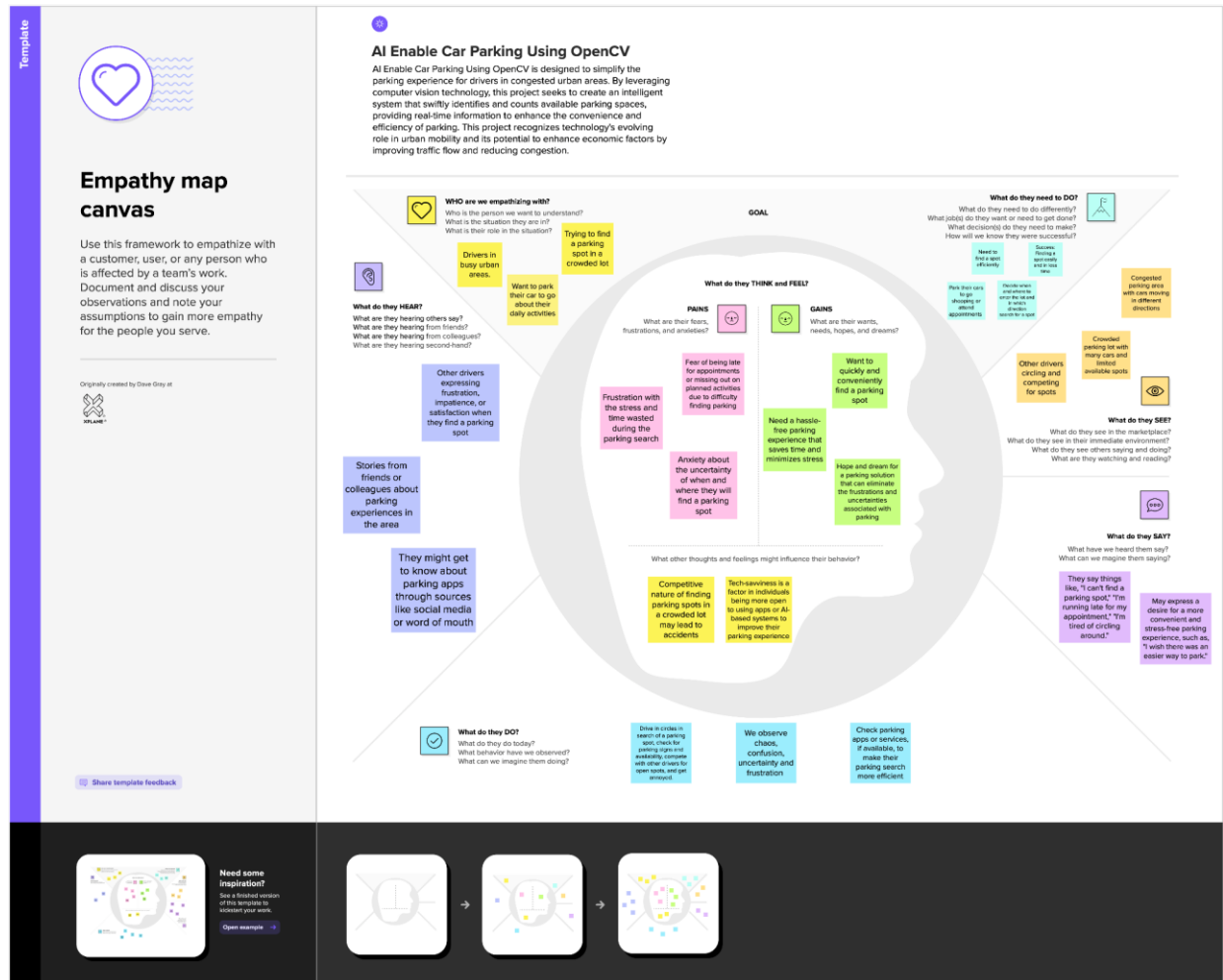
Empathize & Discover

Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes.

It is a useful tool to better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps us consider things from the user's perspective along with his or her goals and challenges.

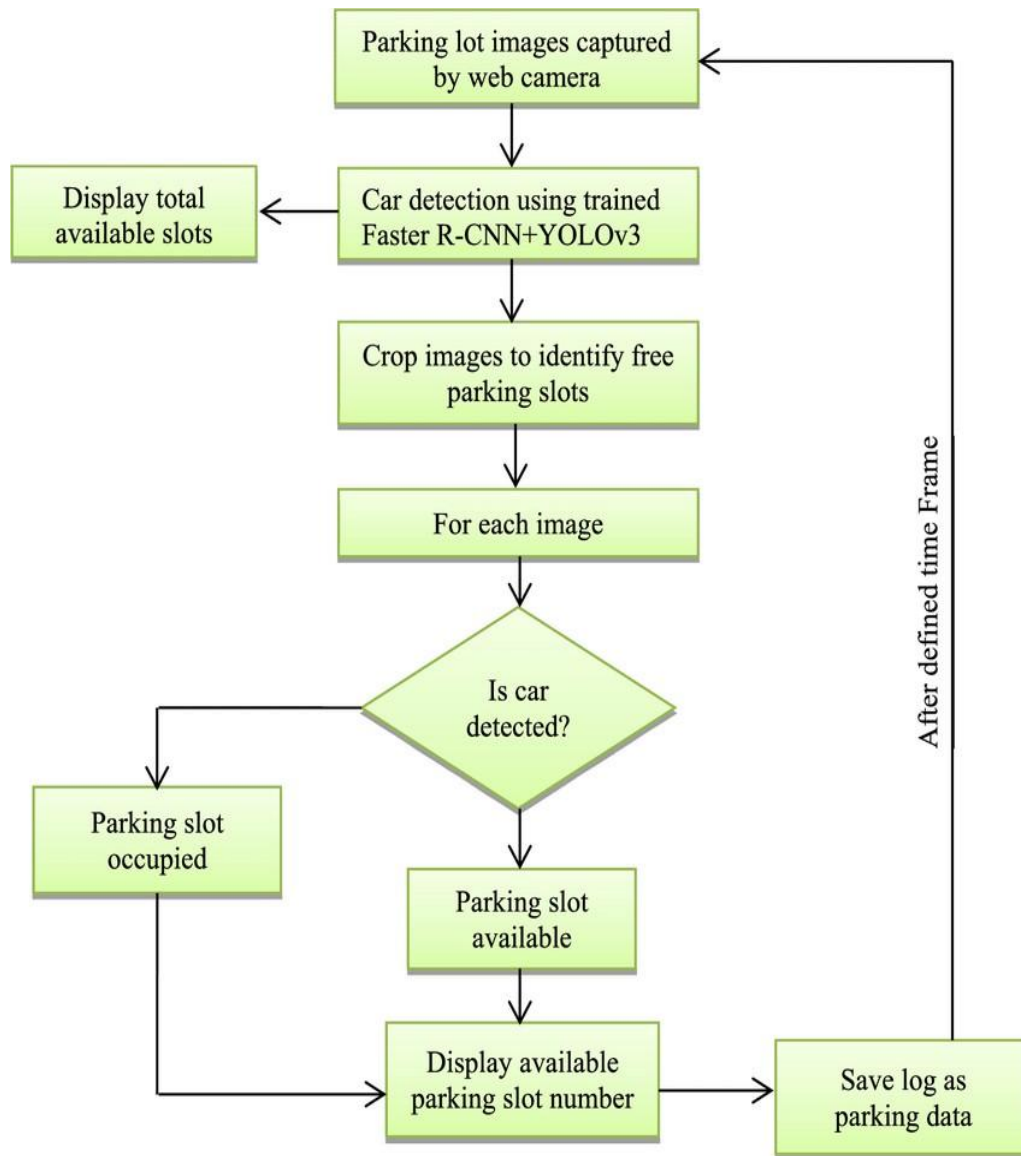


Project Design Phase-II

Data Flow Diagram & User Stories

Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored. Here is the Data flow Diagram of AI Enabled Car Parking using OpenCV from gathering data to deployment.



User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Car Parking Operators and Service Providers	Project setup & Infrastructure	USN-1	Set up the development environment with the required tools and frameworks to develop and deploy the AI-enabled car parking solution using OpenCV.	Successfully configure the development environment with all necessary tools and frameworks.	High	Sprint 1
Municipalities and Local Governments	Data collection	USN-2	Gather a diverse dataset of parking lot images captured via cameras in various urban settings to train the AI model effectively.	Collect a diverse dataset with images of different parking scenarios, including crowded lots, empty lots, and various lighting conditions.	High	Sprint 1
Project	Data preprocessing	USN-3	Preprocess the collected dataset by resizing images, normalizing pixel values, and splitting it into training and validation sets.	Successfully preprocess the dataset, ensuring it's ready for model training.	High	Sprint 2
Project	Model development	USN-4	Explore and evaluate different computer vision models using OpenCV for parking spot detection and classification to determine the most suitable model for the project.	Experiment with various OpenCV-based models to choose the most accurate and efficient one.	High	Sprint 2
Project	Model development	USN-5	Train the selected model using the preprocessed dataset and monitor its performance on the validation set.	Train the model and verify its accuracy and effectiveness in detecting empty parking spots.	High	Sprint 3

Project	Model development	USN-6	Implement real-time video processing for continuous parking spot detection and tracking using OpenCV.	Successfully process live video footage to identify and count available parking spots.	Medium	Sprint 3
Project	Model deployment & Integration	USN-7	Develop a Flask-based web application to deploy the trained model. Integrate the model's output into the web app, allowing users to access the available parking spot count through a user-friendly interface.	Successfully develop and deploy a web application using Flask that integrates the trained model and provides parking spot information to users.	Medium	Sprint 4
Testing & quality assurance	Testing & quality assurance	USN-8	Conduct extensive testing and quality assurance of the model and the web application to identify and report any issues, bugs, or inaccuracies. Optimize model hyperparameters based on user feedback and testing results	Conduct thorough testing, ensure the model's accuracy, and fine-tune it based on user feedback and testing results.	Medium	Sprint 5

Project Design Phase-I

Solution Architecture

Solution Architecture:

1. Data Gathering:

- Places cameras strategically within the parking lot to capture a live video feed.
- Ensures continuous collection of video data for a real-time view of the parking lot.

2. Image Preprocessing:

- Preprocesses each frame from the video feed to enhance image quality and extract key features.
- Utilizes techniques like resizing, noise reduction, and contrast adjustment to improve image clarity.

3. Model Building:

- Develops a robust machine learning model, potentially employing convolutional neural networks (CNNs), to analyze preprocessed images.
- Trains the model on labeled data to recognize parking spaces and accurately identify unoccupied spots.

4. Empty Parking Space Detection:

- Systematically processes each frame using the model to identify parking spaces.
- Classifies parking spaces as either 'occupied' or 'empty' based on the presence of vehicles.
- Utilizes object detection algorithms to precisely locate vehicles within parking spots.

5. Real-Time Analysis:

- Provides continuous, real-time analysis of the live video feed, actively monitoring parking space status.
- Updates the count of available parking spots in real-time.

6. User Interface:

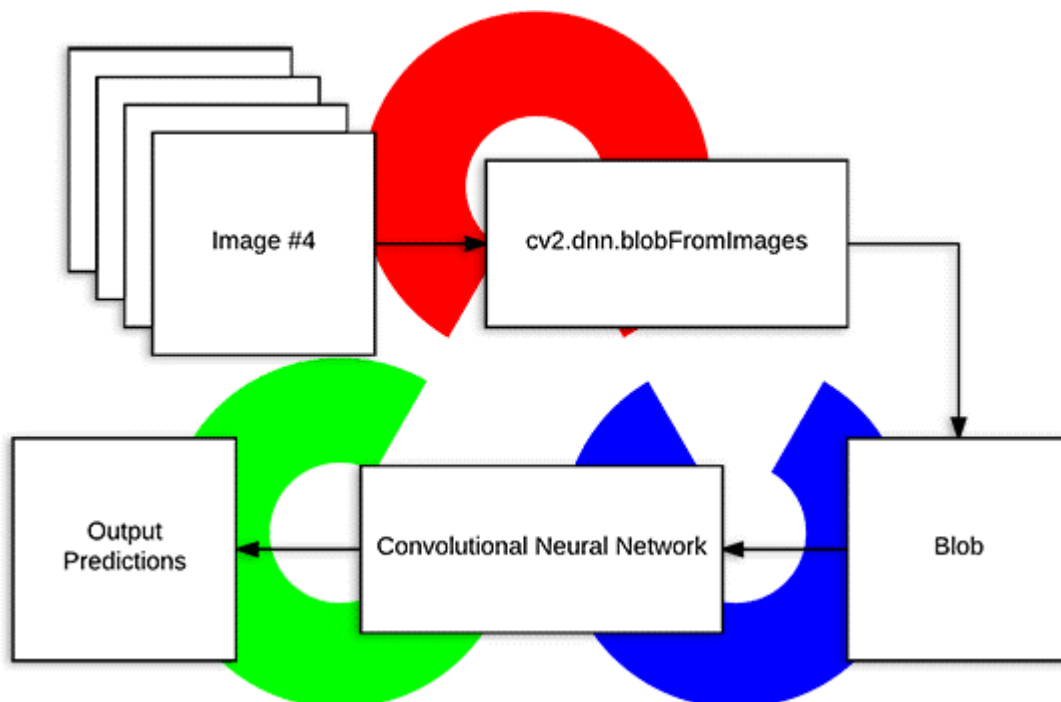
- Offers a user-friendly interface, such as a mobile app or a digital display at the parking lot entrance.
- Provides an up-to-the-minute count of available parking spots, enabling drivers to quickly locate parking spaces.

7. Alerts and Notifications:

- Generates alerts or notifications for parking management personnel and drivers based on specific conditions, such as the parking lot reaching a certain capacity or detecting unauthorized parking.

In summary, this AI-enabled car parking solution employs cutting-edge technology to revolutionize the parking experience, ensuring efficiency, user-friendliness, and real-time monitoring for enhanced overall parking management.

Example - Solution Architecture Diagram:

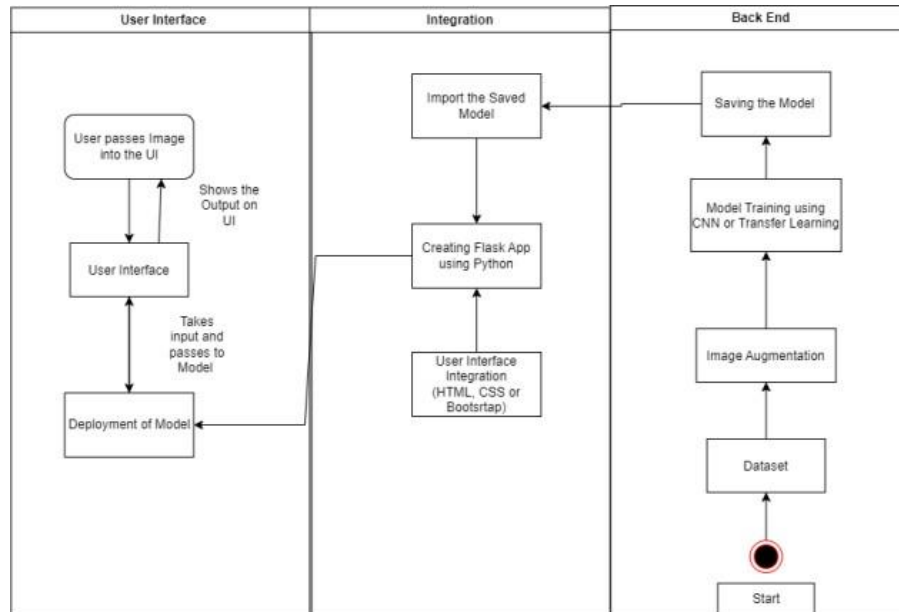


Project Design Phase-II

Technology Stack (Architecture & Stack)

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table2



Guidelines:

1. Include all the processes (As an application logic / Technology Block)
2. Provide infrastructural demarcation (Local / Cloud)
3. Indicate external interfaces (third party API's etc.)
4. Indicate Data Storage components / services
5. Indicate interface to machine learning models (if applicable)

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	Logic for a process in the application	Python
3.	Database	Collect the Dataset Based on the Problem Statement	File Manager
4.	File Storage/ Data	File storage requirements for Storing the dataset	Local System, Google Drive
5.	Frame Work	Used to Create a web Application, Integrating Frontend and Back End	Python Flask
6.	Deep Learning Model	Purpose of Model	OpenCV, CNN
7.	Infrastructure	Application Deployment on Local System	Local Server

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Python's Flask or any basic web framework	Python Flask
2.	Security Implementations	Basic encryption for data transmission, simple access controls	Basic Encryption (e.g., HTTPS), Basic Access Control
3.	Scalable Architecture	Simple, non-distributed architecture	Single-Server Setup

S.No	Characteristics	Description	Technology
4.	Availability	Local server setup for development and testing	Local Server
5.	Performance	Basic performance considerations for local use	Basic Caching Strategies, Local Testing

Project Planning Phase
Project Planning Template (Product Backlog, Sprint Planning, Stories, Story points)

Product Backlog, Sprint Schedule, and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	Project setup & Infrastructure	USN-1	Install and configure cameras and hardware components in the parking lot for real-time video feed.	1	High
Sprint-1	Development environment	USN-2	Set up the development environment with the required tools and frameworks to start the car parking project	1	High
Sprint-2	Object Detection	USN-3	Develop an object detection model to identify cars in real-time using OpenCV and a pre-trained CNN architecture (e.g., YOLO)	5	High
Sprint-2	Parking Space Detection	USN-4	Implement a parking space detection model to identify vacant and occupied parking spaces using OpenCV	6	High

Sprint-3	Real-time Parking Lot Occupancy Detection	USN-5	Integrate the object detection and parking space detection models into the OpenCV pipeline for real-time analysis of parking lot occupancy	6	High
Sprint-3	Updating model to keep count of vacant parking spaces	USN-6	Develop an algorithm to maintain a count of available parking spaces based on real-time data	3	Medium
Sprint-4	Testing & quality assurance	USN-7	Conduct extensive testing under different lighting and weather conditions to ensure system accuracy	5	Medium
Sprint-5	Model deployment and integration	USN-8	Deploy the AI-enabled car parking system in a real-world parking lot and monitor its performance.	4	Medium

AI ENABLED CAR USING OPENCV

Introduction:

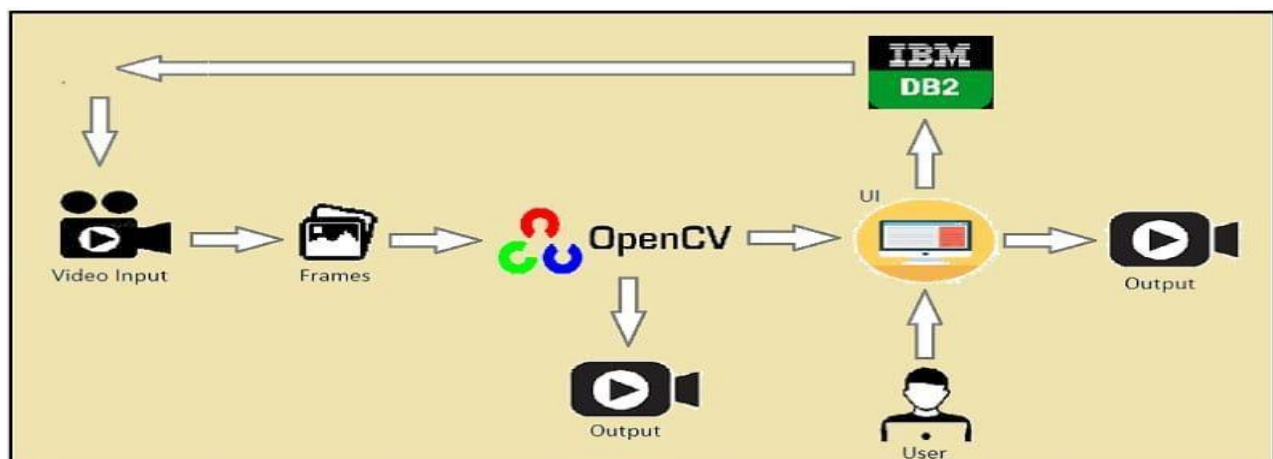
In the coming few years, the surge in urbanization is anticipated to lead to an increased demand for parking facilities, driven by the growing urban population and the continuous expansion of car ownership. This trend poses a considerable challenge in efficiently managing parking spaces. Traditional methods of parking management, such as manual attendants or static signage, are proving to be inadequate and outdated in the face of this rapid urbanization.

To tackle this emerging challenge and leverage the capabilities of modern technology, we are introducing an AI-enabled car parking system that incorporates OpenCV and advanced artificial intelligence algorithms. The primary objective of this system is to revolutionize the parking experience by enhancing convenience, efficiency, and user-friendliness.

Similar to the waste separation system you described as a response to the challenges of waste accumulation, our AI-enabled car parking system provides a smarter solution to the pressing issues in parking. By integrating computer vision and AI technologies, the system automates the entire parking process, from identifying vehicles to allocating parking spaces. This means an end to the frustration of searching for an available spot in a parking lot or dealing with human attendants.

The core functionality of the system is reliant on OpenCV, a robust open-source computer vision library, and sophisticated Convolutional Neural Networks (CNNs). It excels in accurately detecting and recognizing vehicles, evaluating available parking spaces, and guiding drivers to their designated spots, thereby reducing both parking time and frustration. Moreover, the system's adaptability makes it suitable for deployment in various environments, encompassing both indoor and outdoor parking facilities.

Solution Architecture:



Prerequisites:

To complete this project, we must require the following software's, concepts, and packages

To successfully undertake the development of an AI-enabled car parking system using OpenCV, you will need the following software, concepts, and packages:

1. **Google Colab:** Google Colab is a free, cloud-based platform that provides a Jupyter notebook environment with access to powerful GPU and CPU resources. This is where you'll be coding and running your AI-enabled car parking project. You can access Google Colab through a web browser.
2. **OpenCV (Open Source Computer Vision Library):** OpenCV is an essential computer vision library that you can use to process and analyze visual data. It provides a wide range of tools for image and video processing, making it a fundamental component of this project.
3. **Python Programming Language:** A good understanding of Python is required, as it's the primary programming language used in this project. Python is known for its simplicity and readability, making it an excellent choice for AI and computer vision projects.
4. **Machine Learning and Computer Vision Concepts:** Familiarity with fundamental machine learning concepts, as well as computer vision techniques, is beneficial. Understanding concepts like object detection, image classification, and image processing will be essential for implementing the AI-enabled car parking system.
5. **OpenCV-Python:** You will need to install the OpenCV-Python package to access OpenCV's functionalities in your Google Colab environment. You can install it using pip or the package manager provided by Google Colab.

1. To build Machine learning models we require the following packages

- **Numpy:**
 - It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to perform mathematical operations
- **Scikit-learn:**
 - It is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and SciPy
- **Flask:**

Web framework used for building Web applications
- **Python packages:**
 - open anaconda prompt as administrator
 - Type “pip install numpy” and click enter.
 - Type “pip install pandas” and click enter.
 - Type “pip install scikit-learn” and click enter.
 - Type “pip install tensorflow==2.3.2” and click enter.
 - Type “pip install keras==2.3.1” and click enter

Deep Learning Concepts

- **CNN:** a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery.

CNN Basic

- **Flask:** Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications.

Flask Basics

Project Objectives:

By the end of this project you will:

- Know fundamental concepts and techniques of Convolutional Neural Network.
- Gain a broad understanding of image data.
- Know how to pre-process/clean the data using different data preprocessing techniques.
- know how to build a web application using the Flask framework.

Project Flow:

- The user interacts with the UI (User Interface) to choose the image.
- The chosen image analyzed by the model which is integrated with flask application.
- CNN Models analyze the image, then prediction is showcased on the Flask UI.

To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
 - Create Train and Test Folders.
- Data Preprocessing.
 - Import the ImageDataGenerator library
 - Configure ImageDataGenerator class
 - Apply ImageDataGenerator functionality to Trainset and Testset
- Model Building
 - Import the model building Libraries
 - Initializing the model
 - Adding Input Layer
 - Adding Output Layer
 - Configure the Learning Process
 - Training and testing the model
 - Save the Model
 - Application Building
 - Create an HTML file
 - Build Python Code

Milestone 1: Data Collection

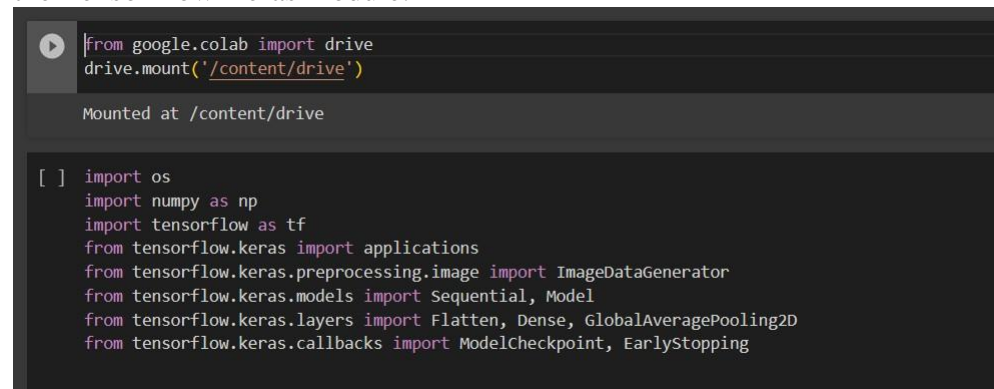
We received a single image and transformed it into sections representing occupied and vacant spaces to minimize plagiarism.

Milestone 2: Image Preprocessing

In this milestone we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although perform some geometric transformations of images like rotation, scaling, translation, etc.

Activity 1: Import the ImageDataGenerator library

Utilizing image data augmentation is a method to artificially increase the scale of a training dataset by generating altered versions of images within the dataset. The Keras deep learning neural network library offers the functionality to train models using image data augmentation through the implementation of the ImageDataGenerator class. Let's import the ImageDataGenerator class from the TensorFlow Keras module.



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

Mounted at /content/drive

[ ] import os
import numpy as np
import tensorflow as tf
from tensorflow.keras import applications
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Flatten, Dense, GlobalAveragePooling2D
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
```


Activity 2: Configure ImageDataGenerator class

The ImageDataGenerator class is instantiated to configure various types of data augmentation techniques. These techniques include:

- Image shifts using the width_shift_range and height_shift_range parameters.
- Image flips through the horizontal_flip and vertical_flip parameters.
- Image rotations specified by the rotation_range parameter.
- Image brightness adjustments controlled by the brightness_range parameter.
- Image zooming defined by the zoom_range parameter.

To apply these augmentation techniques separately for training and testing, instances of the ImageDataGenerator class can be created.

Image Data Augmentation

```
# Data augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    horizontal_flip=True,
    fill_mode="nearest",
    zoom_range=0.1,
    width_shift_range=0.1,
    height_shift_range=0.1,
    rotation_range=5)

test_datagen = ImageDataGenerator(
    rescale=1./255,
    horizontal_flip=True,
    fill_mode="nearest",
    zoom_range=0.1,
    width_shift_range=0.1,
    height_shift_range=0.1,
    rotation_range=5)
```

Activity 3: Apply ImageDataGenerator functionality to Trainset and Testset

The ImageDataGenerator class is instantiated to configure various types of data augmentation techniques. These techniques include:

- Image shifts using the width_shift_range and height_shift_range parameters.
- Image flips through the horizontal_flip and vertical_flip parameters.
- Image rotations specified by the rotation_range parameter.
- Image brightness adjustments controlled by the brightness_range parameter.
- Image zooming defined by the zoom_range parameter.

To apply these augmentation techniques separately for training and testing, instances of the ImageDataGenerator class can be created. **Loading Our Data And Performing Data Augmentation**

```
[ ] # Creating data generators
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode="categorical")

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_height, img_width),
    class_mode="categorical")

Found 381 images belonging to 2 classes.
Found 164 images belonging to 2 classes.
```

We notice that 381 images belong to 2 classes for training and 164 images belong to 2 classes for testing purposes.

Milestone 3: Model Building

Now it's time to build our Convolutional Neural Networking which contains an input layer along with the convolution, max-pooling, and finally an output layer.

Activity 1: Importing the Model Building Libraries

```
[ ] import os
import numpy as np
import tensorflow as tf
from tensorflow.keras import applications
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Flatten, Dense, GlobalAveragePooling2D
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
```

Activity 2: Initializing the model

```
[ ] # Loading pre-trained VGG16 model
    base_model = applications.VGG16(weights="imagenet", include_top=False, input_shape=(img_width, img_height, 3))

    # Freezing layers in the base model
    for layer in base_model.layers[:10]:
        layer.trainable = False

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16\_weights\_tf\_dim\_ordering\_tf\_kernels\_notop.h5
58889256/58889256 [=====] - 0s 0us/step
```

Activity 3: Adding Dense Layers

A dense layer is a deeply connected neural network layer. It is the most common and frequently used layer.

```
[ ] # Adding custom dense layer for binary classification
    x = GlobalAveragePooling2D()(base_model.output)
    x = Dense(units=2, activation='softmax')(x)
    model = Model(base_model.input, x)
```

The Dense layer's neuron count aligns with the number of classes in the training set. In the final Dense layer, softmax activation is applied to transform outputs into corresponding probabilities.

Gaining insight into the model is crucial for effective utilization in training and prediction. Keras offers a straightforward 'summary' method to obtain comprehensive information about the model and its individual layers.

Summary of the Model

model.summary()

Model: "model"

Layer (type)	Output Shape	Param #
=====		
input_1 (InputLayer)	[(None, 128, 128, 3)]	0
block1_conv1 (Conv2D)	(None, 128, 128, 64)	1792
block1_conv2 (Conv2D)	(None, 128, 128, 64)	36928
block1_pool (MaxPooling2D)	(None, 64, 64, 64)	0
block2_conv1 (Conv2D)	(None, 64, 64, 128)	73856
block2_conv2 (Conv2D)	(None, 64, 64, 128)	147584
block2_pool (MaxPooling2D)	(None, 32, 32, 128)	0
block3_conv1 (Conv2D)	(None, 32, 32, 256)	295168
block3_conv2 (Conv2D)	(None, 32, 32, 256)	590080
block3_conv3 (Conv2D)	(None, 32, 32, 256)	590080
block3_pool (MaxPooling2D)	(None, 16, 16, 256)	0
block4_conv1 (Conv2D)	(None, 16, 16, 512)	1180160
block4_conv2 (Conv2D)	(None, 16, 16, 512)	2359808
block4_conv3 (Conv2D)	(None, 16, 16, 512)	2359808
block4_pool (MaxPooling2D)	(None, 8, 8, 512)	0
block5_conv1 (Conv2D)	(None, 8, 8, 512)	2359808
block5_conv2 (Conv2D)	(None, 8, 8, 512)	2359808
block5_conv3 (Conv2D)	(None, 8, 8, 512)	2359808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 2)	1026
=====		
Total params: 14715714 (56.14 MB)		
Trainable params: 12980226 (49.52 MB)		
Non-trainable params: 1735488 (6.62 MB)		

Activity 4: Configure The Learning Process

1. The culmination of model creation involves compilation, marking the transition to the training phase. The loss function, integral for identifying learning errors or deviations, is a prerequisite during the model compilation in Keras.
2. Optimization is a critical step in refining input weights through a comparison between predictions and the loss function. In this context, the adam optimizer is employed.
3. Metrics play a role in assessing model performance, akin to the loss function, although they aren't actively involved in the training process.

Compiling the Model

```
[ ] # Model training
checkpoint = ModelCheckpoint("car1.h5", monitor='val_acc', verbose=1, save_best_only=True, save_weights_only=False, mode='auto', period=1)
early = EarlyStopping(monitor='val_acc', min_delta=0, patience=10, verbose=1, mode='auto')

history_object = model.fit_generator(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator,
    callbacks=[checkpoint, early])
```

Activity 5: Train The model

To train our model using the image dataset, the `fit_generator` function is employed. The model undergoes training for 30 epochs, and after each epoch, the current model state is saved if the encountered loss is the lowest up to that point. The training loss exhibits a consistent decrease in almost every epoch throughout the 30 iterations, suggesting potential for further model improvement.

The `fit_generator` function takes several arguments:

- **steps_per_epoch**: This parameter signifies the total number of steps taken from the generator once one epoch is completed and the subsequent epoch begins. The calculation for `steps_per_epoch` involves dividing the total number of samples in the dataset by the batch size.
- **Epochs**: This is an integer representing the number of epochs for which the model is trained.
- **validation_data**: It can be either a list of inputs and targets, a generator, or a list containing inputs, targets, and sample weights. This parameter is used to evaluate the loss and metrics for the model after each epoch.
- **validation_steps**: This argument is applicable only if the `validation_data` is a generator. It specifies the total number of steps taken from the generator before it stops at the end of each epoch. The value of `validation_steps` is calculated by dividing the total number of validation data points in the dataset by the validation batch size.

Fit the Model

```
WARNING:tensorflow:'period' argument is deprecated. Please use 'save_freq' to specify the frequency in number of batches seen.
<ipython-input-11-c9c9e48c8e87>:5: UserWarning: 'Model.fit_generator' is deprecated and will be removed in a future version. Please use 'Model.fit', which supports generators.
  history_object = model.fit_generator(
Epoch 1/5
12/12 [=====] - ETA: 0s - loss: 0.6159 - accuracy: 0.6903 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric 'val_acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
12/12 [=====] - 265s 22s/step - loss: 0.6159 - accuracy: 0.6903 - val_loss: 0.3575 - val_accuracy: 0.8110
Epoch 2/5
12/12 [=====] - ETA: 0s - loss: 0.2686 - accuracy: 0.8635 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric 'val_acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
12/12 [=====] - 287s 17s/step - loss: 0.2686 - accuracy: 0.8635 - val_loss: 0.5826 - val_accuracy: 0.8598
Epoch 3/5
12/12 [=====] - ETA: 0s - loss: 0.2327 - accuracy: 0.9134 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric 'val_acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
12/12 [=====] - 283s 17s/step - loss: 0.2327 - accuracy: 0.9134 - val_loss: 0.3333 - val_accuracy: 0.8841
Epoch 4/5
12/12 [=====] - ETA: 0s - loss: 0.1867 - accuracy: 0.9186 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric 'val_acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
12/12 [=====] - 218s 18s/step - loss: 0.1867 - accuracy: 0.9186 - val_loss: 0.2368 - val_accuracy: 0.8963
Epoch 5/5
12/12 [=====] - ETA: 0s - loss: 0.1423 - accuracy: 0.9423 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Early stopping conditioned on metric 'val_acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy
12/12 [=====] - 209s 18s/step - loss: 0.1423 - accuracy: 0.9423 - val_loss: 0.3345 - val_accuracy: 0.8963
```

Activity 6: Save the Model

The model is saved with .h5 extension as follows

An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

```
[ ] # Save the model as a pickle file
import pickle

model_dict = {'model': model.to_json(), 'weights': model.get_weights()}

with open('/content/drive/MyDrive/AI_Enabled_Car_Parking/spot_dict.pickle', 'wb') as f:
    pickle.dump(model_dict, f)
```

Activity 7: Test The model

Evaluation is an integral phase in model development aimed at assessing whether the model is the optimal solution for the given problem and associated dataset. Load the saved model using `load_model`

```
[ ] # Model prediction function
def prediction(path):
    img = tf.keras.preprocessing.image.load_img(path, target_size=(img_height, img_width))
    img_array = tf.keras.preprocessing.image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0 # Rescaling
    pred = np.argmax(model.predict(img_array))
    print(f"The image belongs to class {pred}")

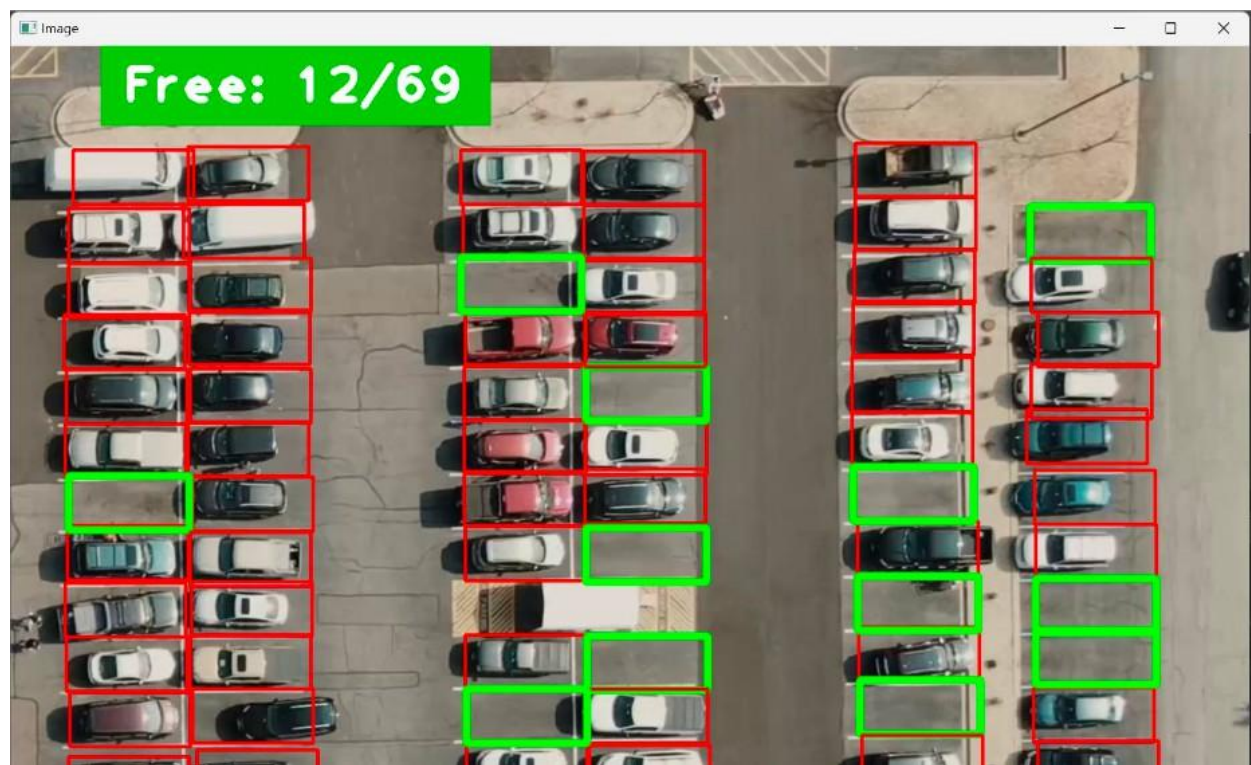
[ ] # Example prediction
image_path = "/content/drive/MyDrive/AI_Enabled_Car_Parking/CarParking_Dataset/test_images/scene1410.jpg"
prediction(image_path)

1/1 [=====] - 1s 628ms/step
The image belongs to class 1
```

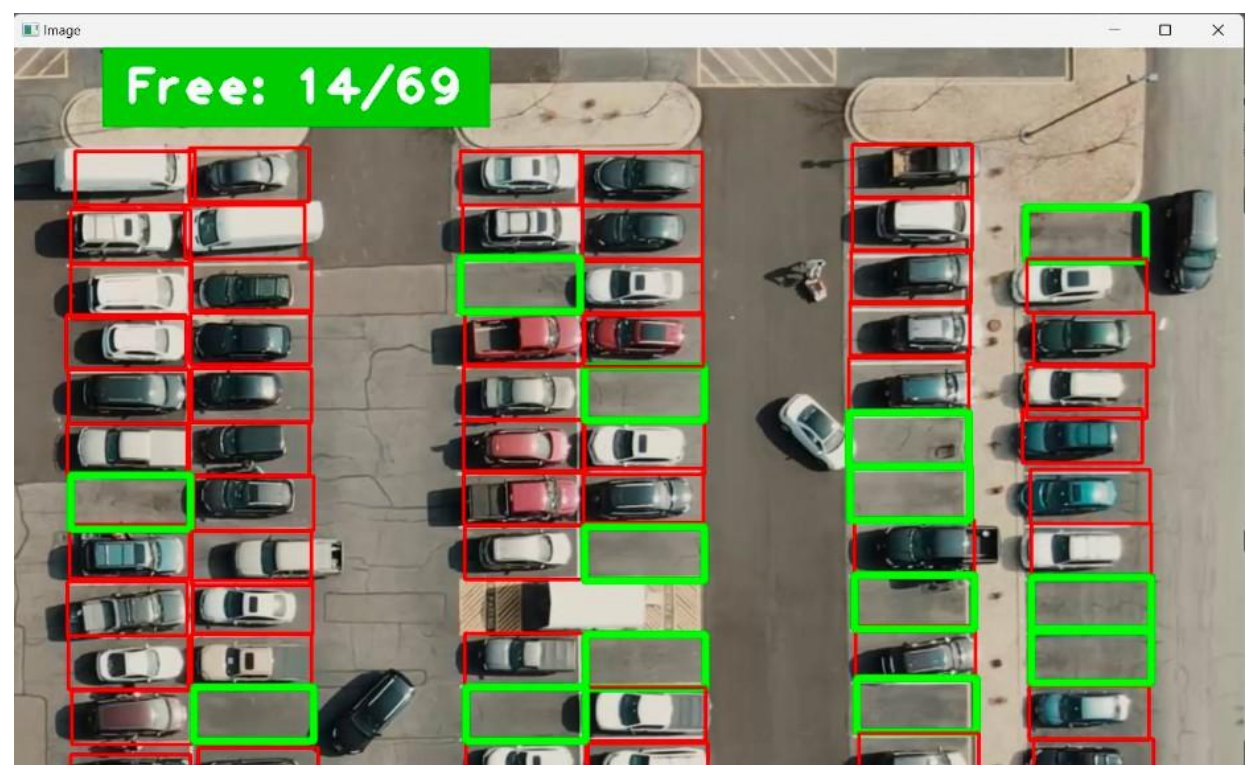

RESULTS:

Screenshots:

1)



2)



Advantages:

1. Cost Optimization:

- Departure from Traditional Models:
- The AI-based car parking system offers an economically viable departure from traditional radar and sensor-based solutions.
- By eliminating the need for expensive hardware installations, it capitalizes on computer vision and AI, thereby significantly reducing initial setup costs.
- The software-driven approach also minimizes maintenance expenses by mitigating the wear and tear usually associated with physical sensors.

2. Enhanced Parking Efficiency:

- Automated Space Detection:
- This system's automated space detection transforms the landscape of parking lot management.
- Real-time identification and allocation of available parking spaces empower drivers to swiftly locate vacant spots, optimizing their parking experience.

3. Traffic Flow Optimization:

- Direct Impact on Congestion:
- The system's proficiency in optimizing parking efficiency translates directly into a reduction in traffic congestion.
- Streamlining the process of finding parking spaces substantially decreases the time vehicles spend circling, promoting smoother traffic flow in urban areas.

4. Environmental Impact Reduction:

- Reduced Emissions:
- Minimizing the time vehicles spend searching for parking directly correlates with reduced fuel consumption and emissions.
- This eco-friendly approach aligns with efforts toward environmental sustainability, showcasing a commitment to mitigating urban pollution.

Disadvantages:

1. Technological Reliance:

- Dependency on Functional Systems:
- The system's operational reliability heavily depends on the consistent and accurate functioning of computer vision and AI systems.
- Any technical disruptions or errors in these systems may compromise the overall reliability and effectiveness of the parking management system.

2. Implementation Costs:

- Initial Investment and Ongoing Expenses:
- Implementing the AI-enabled system may necessitate substantial initial investments in technology, infrastructure, and skilled workforce.
- Sustained expenses in terms of maintenance, upgrades, and incorporation of evolving technologies could amplify the overall costs over time.

3. Privacy and Data Concerns:

- Data Collection and Safeguarding Privacy:
- The collection of data for efficient parking management may invoke privacy concerns among users.
- Implementing rigorous privacy measures and transparent data handling protocols becomes imperative to address these concerns effectively.

Conclusion

The AI-enabled car parking system stands as a promising solution for the efficient management of urban parking spaces. Its array of advantages, encompassing heightened efficiency, diminished traffic congestion, and noteworthy environmental benefits, underscore the technology's positive impact. However, pivotal considerations around technological dependency, implementation expenses, and privacy concerns necessitate careful attention.

The success of this system pivots on striking an equilibrium between technological innovation and the effective resolution of associated challenges. While the system brings about remarkable advancements in parking management, addressing these concerns becomes imperative for its seamless integration and sustained effectiveness within urban landscapes. Achieving this balance ensures not only the optimization of parking resources but also cultivates trust and reliability among users, reinforcing its role as an indispensable solution in modern urban infrastructures.

Future Scope

The future trajectory of the AI-enabled car parking system encompasses a continuous journey of advancements and enhancements. Potential areas for future development include:

1. Synergistic Integration with Smart Cities:

- Forge collaborative alliances with smart city initiatives, aiming for a seamless fusion into the fabric of urban infrastructure.
- Leverage interconnectivity to optimize traffic management and overall urban mobility.

2. Augmented AI Capabilities:

- Elevate AI algorithms to achieve heightened precision in parking space detection, ensuring even greater accuracy.
- Dive deeper into machine learning methodologies to perpetually refine and enhance the system's functionalities.

3. Intuitive User Interfaces:

- Pioneering the development of user-friendly mobile applications, designed for effortless access to comprehensive parking information.
- Implementation of responsive user feedback mechanisms to foster continuous system optimization based on user experiences and preferences.

4. Fortified Security and Privacy Measures:

- Channel resources into fortifying robust security measures, safeguarding data integrity and user privacy.
- Regularly update protocols and encryption technologies to proactively address potential security vulnerabilities.

5. Global Proliferation:

- Spearhead global advocacy campaigns to encourage the widespread adoption of AI-enabled parking systems.
- Foster collaborations with governmental bodies, municipalities, and private entities, facilitating expansive implementation across diverse geographic regions.

This vision for future enhancements aims not only to refine the technological prowess of AI-enabled parking systems but also emphasizes user-centric approaches, security fortification, and global proliferation, ensuring its seamless integration and long-term sustainability in diverse urban landscapes worldwide.