**A Project Report on**

**Guidelines for the Preparation of Project Thesis**

**Department of CSE**

submitted in partial fulfillment for the award of

**Bachelor of Technology**

in

**Computer Science & Engineering**

by

**P Lakshmi Sahithi (Y20ACS533) P Hanumanth Rao(L21ACS414)**

**P Vyshnavi (Y20ACS537) Sk Asifa(Y20ACS561)**



Under the guidance of

**Mr.T.Y.Srinivasa Rao**

Department of Computer Science and Engineering

**Bapatla Engineering College**

(Autonomous)

(Affiliated to Acharya Nagarjuna University)

**BAPATLA – 522 102, Andhra Pradesh, INDIA**

**2023-2024**

**Department of**

**Computer Science & Engineering**



**CERTIFICATE**

This is to certify that the project report entitled **Guidelines for the Preparation of Project Thesis Department of CSE** that is being submitted by P Lakshmi Sahithi (Y20ACS533), P Hanumanth Rao(L21ACS414), P Vyshnavi(Y20ACS537) and Sk .Asifa (Y20ACS561) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

**Signature of the Guide Signature of the HOD**

**Mr.T.Y.Srinivasa Rao Dr.M.Rajesh Babu**

**Assistant Professor Prof. & HeadDECLARATION**

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

**P Lakshmi Sahithi (Y20ACS533)**

**P Hanumantha Rao(L21ACS414)**

**P Vyshnavi (Y20ACS537)**

**Sk Asifa(y20ACS561)**

**Acknowledgement**

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **Mr.T.Y.Srinivasa Rao,Assistant professor,** Department of CSE, for his/her valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr.M.Rajesh Babu**, Prof. & Head of the Dept. for extending his cooperation and providing the required resources.

We would like to thank our beloved Principal **Dr.Nazeer Shaik**  for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Dr. N. Sudhakar,** Prof. Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

**P Lakshmi Sahithi (Y20ACS533)**

**P Hanumantha Rao (L21ACS414)**

**P Vyshnavi (Y20ACS537)**

**Sk Asifa (y20ACS561)**

**ABSTRACT**

The escalating demand for parking spaces in urban areas has spurred challenges in parking management, including traffic congestion and inefficiencies. To mitigate these issues, this study introduces a smart parking system integrating computer vision techniques with machine learning technologies. Through a network of cameras, real-time images of parking areas are captured and analyzed using computer vision methods to identify and segment parking spaces. Subsequently, machine learning algorithms, notably a Support Vector Machine (SVM) classifier, categorize each space as vacant or occupied based on extracted features. This facilitates the provision of real-time information on parking space availability, enabling efficient parking management and optimization while enhancing the overall urban mobility experience.

The system's compatibility with digital displays enables seamless integration with existing urban infrastructure, offering dynamic updates on parking availability. By amalgamating computer vision and machine learning, the proposed smart parking system offers a comprehensive solution to urban parking challenges. It addresses the growing demand for parking spaces, reduces congestion, and improves urban mobility, contributing to a more sustainable and efficient urban environment.

Table of Contents

[**ABSTRACT** iv](#_Toc164323048)

[Table of Contents v](#_Toc164323049)

[**List of figures** vii](#_Toc164323050)

[1 Introduction 1](#_Toc164323051)

[1.1 Machine Learning 1](#_Toc164323052)

[1.1.1 Types of Machine Learning 2](#_Toc164323053)

[1.1.2 Machine Learning vs Traditional Programming 10](#_Toc164323054)

[1.1.3 How Does Machine Learning Work? 11](#_Toc164323055)

[1.1.4 Challenges and Limitations of Machine Learning 13](#_Toc164323056)

[1.2 Introduction to Computer Vision 15](#_Toc164323057)

[1.2.1 How Computer Vision Works? 16](#_Toc164323058)

[1.2.2 Applications of Computer Vision 18](#_Toc164323059)

[1.2.3 Fundamental of Image Processing 19](#_Toc164323060)

[1.2.4 Computer Vision Techinques 21](#_Toc164323061)

[1.2.5 Challenges and Future Directions in Computer Vision 22](#_Toc164323062)

[1.3 Motivation 24](#_Toc164323063)

[1.4 Problem Statement 24](#_Toc164323064)

[1.5 Scope of The Project 25](#_Toc164323065)

[2 Literature Survey 26](#_Toc164323066)

[3 System Analysis 28](#_Toc164323067)

[3.1 Existing System 28](#_Toc164323068)

[3.1.1 Methodology 28](#_Toc164323069)

[3.1.2 Drawbacks 30](#_Toc164323070)

[3.2 Proposed System 30](#_Toc164323071)

[3.2.1 How Model Trained? 30](#_Toc164323072)

[3.2.2 How Model Works? 33](#_Toc164323073)

[4 Implementation 35](#_Toc164323074)

[4.1 Software Requirements 35](#_Toc164323075)

[4.2 Libraries and APIs Used 36](#_Toc164323076)

[5 System Design 44](#_Toc164323077)

[5.1 Use Case Diagram. 44](#_Toc164323078)

[5.2 Class Diagram 45](#_Toc164323079)

[5.3 Activity Diagram. 46](#_Toc164323080)

[5.4 Sequence Diagram. 47](#_Toc164323081)

[5.5 State Chart Diagram 47](#_Toc164323082)

[6 Result 49](#_Toc164323083)

[7 Conclusion 51](#_Toc164323084)

[8 Bibliography 53](#_Toc164323085)

**List of figures**

[Figure 1.1 Types of Machine Learning 9](#_Toc164291577)

[Figure 1.2 Traditional Programming 17](#_Toc164291578)

[Figure 1.3 Machine Learning 17](#_Toc164291579)

[Figure 1.4 Learning Phase 18](#_Toc164291580)

[Figure 1.5 Inference from Model 19](#_Toc164291581)

[Figure 1.6 Human Vision vs Computer Vision 22](#_Toc164291582)

[Figure 1.7 Computer Vision Application 24](#_Toc164291583)

[Figure 3.1 Empty Dataset 37](#_Toc164291584)

[Figure 3.2 Not\_Empty Dataset 37](#_Toc164291585)

[Figure 3.3 Classification Metrics 39](#_Toc164291586)

[Figure 5.1 Usecase Diagram 50](#_Toc164291587)

[Figure 5.2 Class Diagram 51](#_Toc164291588)

[Figure 5.3 Activity diagram 52](#_Toc164291589)

[Figure 5.4 Sequence Diagram 53](#_Toc164291590)

[Figure 5.5 StateChart Diagram 54](#_Toc164291591)

[Figure 6.1 Home Page 56](#_Toc164291592)

[Figure 6.2 Upload Page 56](#_Toc164291593)

[Figure 6.3 Result Page 56](#_Toc164291594)

# Introduction

Our project aims to address the pressing need for efficient parking space management in urban areas using a Support Vector Machine (SVM) classifier. By leveraging fundamental computer vision and machine learning techniques, our goal is to provide a reliable method for real-time parking space detection. Through the analysis of video footage from parking areas, the SVM classifier identifies individual parking spots and determines their occupancy status. This information is then used to optimize parking management and improve urban mobility. With a focus on simplicity and effectiveness, our solution aims to meet the increasing demand for efficient parking space management in urban environments.

## Machine Learning

Machine Learning involves the development of algorithms that enable computers to learn from data and make predictions or decisions without explicit programming. By analyzing large datasets, these algorithms identify patterns and relationships, allowing for accurate predictions on unseen data.

Through statistical techniques and computational algorithms, Machine Learning empowers computers to extract meaningful insights from data, enabling applications like recommendation systems, image recognition, and natural language processing. This transformative technology revolutionizes decision-making processes across various industries.

### Types of Machine Learning

Based on the methods and way of learning, machine learning is divided into mainly four types, which are:

1. Supervised Machine Learning.

2. Unsupervised Machine Learning.

3. Reinforcement Learning

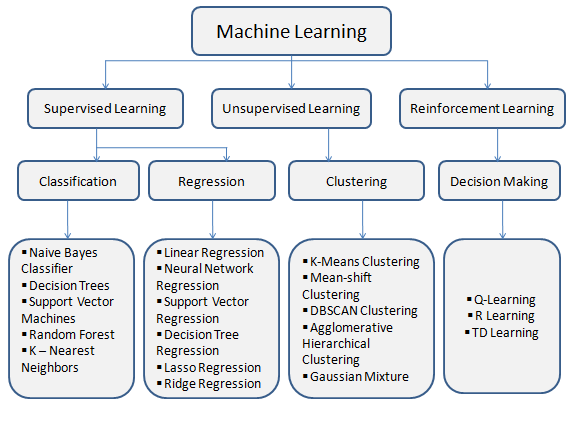


Figure 1.1 Types of Machine Learning

**1.Supervised Machine Learning:**

It uses “Labelled data” to train the model. “Labelled data” means “the output is already known to us”. The model, just needs to map the labelled inputs to the known outputs. The machine knows the features of the data and also the labels associated with features.

Here is how a supervised algorithm works:

a. You feed it an example input, then the associated output.

b. You repeat the above step many times. Eventually, the algorithm picks up a pattern between the inputs and outputs.

c. Now, you can feed it a new input, and it will predict the output for you.

**Training Process:**

Supervised learning algorithms need external supervision to train the models. They need guidance and additional information to return the result.

**Types of problems can be solved:**

Supervised learning algorithms are generally used to solve “Classification and Regression problems”.

**Applications of Supervised learning:**

a. Weather Prediction. (For a particular day based on the humidity, precipitation, wind speed and pressure values)

b. Sales Forecast. (Forecast sales for the next month/quarter for different products)

c. Stock Price Analysis.

d. Risk Assessment. (To assess risk in financial services)

e. Image Classification.

**The following are the commonly used algorithms in supervised learning:**

a. Linear Regression.

b. Logistic Regression.

c. Support Vector Machines.

d. K Nearest Neighbours.

**Eg:** Suppose your friend gives you 1 million coins of 3 different currencies. Each coin has different weights (1 Rupee = 3 grams, 1 Euro = 7 grams, 1 Cent = 4 grams). Machine Learning model will predict the currency of a coin. In this example, “Weight” becomes the “Feature” & “Currency” becomes the “Label”. When we feed this data to the Machine Learning model, it learns “which feature is associated with which label”. If we give a new coin to the machine, on the basis of the weight, Machine Learning model will predict the currency of the given coin.

**Types of Supervised Machine Learning:**

**a. Classification:**

In classification, the output variable is categorical. (yes/no, true/false, etc)

Eg: Predicting an email is spam or not? We need to teach our machine how a spam email looks like, based on spam filters. Spam filters will review the content of an email, email header, keywords, etc. Spam filters will generate a score for an email. The machine decides if it is spam or not based on score. If the score is less, the email is genuine. Otherwise it is considered as spam

**Algorithm**

**Support Vector Machine**

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for both classification and regression tasks. It works by finding the optimal hyperplane (or set of hyperplanes) that best separates the classes in the feature space. SVM is effective in high-dimensional spaces and is particularly well-suited for tasks where the number of dimensions exceeds the number of samples.

Support Vector Classifier (SVC) is the classification variant of SVM. SVC is used specifically for classification tasks, where the goal is to assign a class label to input data points. It works by finding the hyperplane that best separates the classes while maximizing the margin between them. SVC can handle both linearly separable and non-linearly separable datasets using different kernel functions.

**When to Use SVC:**

**1. Binary Classification:**

* SVC is commonly used for binary classification tasks, where the goal is to classify data points into one of two classes.
* Examples include spam detection, sentiment analysis (positive/negative), medical diagnosis (disease/no disease),parking space(empty/non\_empty) etc

**2. Linear Separable Data:**

* SVC is suitable for linearly separable datasets, where classes can be separated by a straight line (or hyperplane in higher dimensions).
* If the data is linearly separable, SVC can find an optimal hyperplane to classify the data accurately.

**3. Non-linearly Separable Data with Kernel Trick :**

* SVC with kernel functions (e.g., polynomial, radial basis function) can handle non-linearly separable datasets.
* The kernel trick allows SVC to map the input data into a higher-dimensional feature space where the classes become linearly separable.

**4. Small to Medium-sized Dataset :**

* SVC works well with small to medium-sized datasets. It can handle datasets with a moderate number of samples and features efficiently.

**5. When Interpretability is Less Important:**

* SVC tends to create complex decision boundaries, especially with non-linear kernel functions.
* If interpretability of the model's decision boundary is less important compared to classification accuracy, SVC can be a suitable choice.

In summary, SVC in SVM is a versatile algorithm suitable for binary classification tasks, especially when dealing with linearly separable or non-linearly separable data. It is effective in a wide range of applications and can handle both small and medium-sized datasets efficiently.

**b.Regression:**

In regression, the output variable is real/continuous value. (Salary based on experience, Weight based on height, etc)

Eg: Let say we have 2 variables (i.e) Temperature and Humidity. Temperature is the independent variable and Humidity is the dependent variable. As the temperature increases, humidity decreases. When we feed this data to the machine, it will understand the relationship between 2 variables. After the machine is trained, it can easily predict the humidity based on the given temperature.

**2. Unsupervised Machine Learning:**

It uses “Unlabeled data” to train the model. “Unlabeled data” means “there is no fixed output variable”. The model learns from the data and understand/discovers patterns and features in the data to discover the output. So, the model tries to label the data based on the patterns/features of the input data.

**Here is how an unsupervised algorithm works:**

a. You feed it an example input (without the associated output).

b. You repeat the above step many times. Eventually, the algorithm clusters your inputs into groups.

c. Now, you can feed it a new input, and the algorithm will predict which cluster it belongs to.

**Training Process:**

Unsupervised learning algorithms do not need any external supervision to train the models. No training will be given to the model. They learn on their own to predict the output. Types of problems can be solved: Unsupervised learning algorithms are generally used to solve “Clustering and Association problems”.

**Applications of Unsupervised learning:**

a. Customer Segmentation. (Based on customer behaviour, likes, dislikes and interests we can cluster similar customers into groups)

b. Customer Churn Analysis.

c. Market Basket Analysis.

d. Semantic Clustering.

**The following are the commonly used algorithms in unsupervised learning:**

a. K Means Clustering.

b. Hierarchical Clustering.

c. DBSCAN.

d. Principal Component Analysis

**Eg:**Suppose we have a cricket dataset of various players with their respective scores and wickets taken. When we feed this dataset to the machine. The machine identifies the “Patterns” of player performance. So, it plots the data as wickets in x axis and runs in y axis. While looking at the data, we can clearly identify that there are 2 clusters (The players who score high runs and took less wickets- batsmen, the players who took more wickets and score less runs- bowler).

**Types of UnSupervised Machine Learning:**

**Clustering**: In clustering, the machine will divide the given input into groups based on the patterns/behavior. Eg: Which customer made similar product purchases?

**4.Reinforcement Machine Learning:**

It is a reward based/feedback based learning. It uses an agent and an environment to produce actions and rewards. On accomplishing a task, the agent receives a reward. In this learning, there is no predefined target variable. It follows “trial and error method” to get the desired solution.

**Training Process:**

Reinforcement learning algorithms do not need any external supervision to train the models just like unsupervised. They learn from the feedback to predict the correct output.

**Types of problems can be solved:**

Reinforcement learning algorithms are generally used to solve “Reward based problems”.

**Applications of Reinforcement learning:**

a. Building Games.

b. Training Robots.

**The following are the commonly used algorithms in reinforcement learning:**

a. Q-Learning.

b. Monte Carlo.

c. SARSA.

d. Deep Q Network.

**Eg:** Suppose we provide an image of a dog to the machine/system and ask it to identify it. The system identifies it as a cat. So, we will give a negative feedback to the machine saying that it is a dog. Then the machine will learn from the feedback and it will be able to identify any other image of dog correctly.

### Machine Learning vs Traditional Programming

Traditional programming differs significantly from machine learning. In traditional programming, a programmer code all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.

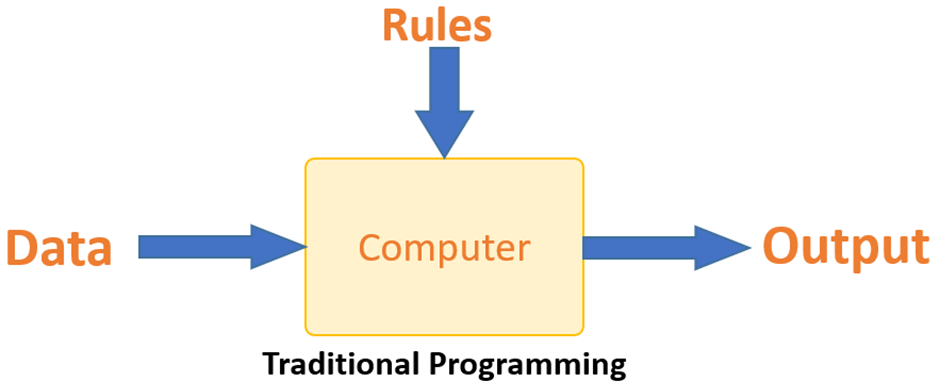


Figure 1.2 Traditional Programming

Machine learning is supposed to overcome this issue. The machine learns how the input and output data are correlated and it writes a rule. The programmers do not need to write new rules each time there is new data. The algorithms adapt in response to new data and experiences to improve efficacy over time.

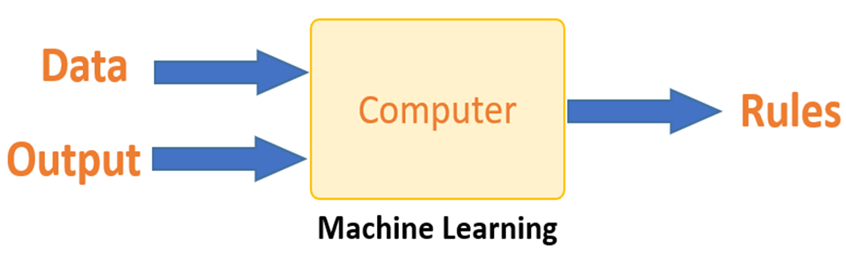


Figure 1.3 Machine Learning

### How Does Machine Learning Work?

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if it’s feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the learning and inference. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the data. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a feature vector. You can think of a feature vector as a subset of data that is used to tackle a problem.

The machine uses some fancy algorithms to simplify the reality and transform this discovery into a model. Therefore, the learning stage is used to describe the data and summarize it into a model.



Figure 1.4 Learning Phase

For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model

**Inferring :**

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.

The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question

2. Collect data

3. Visualize data

4. Train algorithm

5. Test the Algorithm

6. Collect feedback

7. Refine the algorithm

8. Loop 4-7 until the results are satisfying

9. Use the model to make a prediction



Figure 1.5 Inference from Model

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.

### Challenges and Limitations of Machine Learning

Challenges and limitations in machine learning are significant aspects to consider when developing and deploying machine learning systems. Here are some of the key challenges and limitations:

**Data Quality and Quantity:**

**Challenge :** Machine learning models require large amounts of high-quality data for effective training. However, obtaining labeled data can be expensive and time-consuming.

**Limitation :** Limited or poor-quality data can lead to biased or inaccurate models, hindering their performance and reliability.

**Overfitting and Underfitting:**

**Challenge :** Balancing between overfitting (where the model learns noise in the training data and fails to generalize) and underfitting (where the model is too simplistic to capture the underlying patterns) is a common challenge.

**Limitation :** Overfitting can lead to poor performance on unseen data, while underfitting results in models that fail to capture important patterns in the data.

**Interpretability and Explainability:**

**Challenge :** Many machine learning models, particularly deep learning models, are often considered as "black boxes" due to their complexity, making it difficult to understand how they arrive at their predictions.

**Limitation :** Lack of interpretability can be a barrier in critical applications where understanding the reasoning behind predictions is essential (e.g., healthcare or legal decisions).

**Computational Resources:**

**Challenge :** Training complex machine learning models, especially deep learning models, requires significant computational resources, including high-performance GPUs or TPUs.

**Limitation :** Limited access to computational resources can restrict the scalability and speed of training and inference, particularly for resource-intensive models.

**Ethical and Bias Concerns:**

**Challenge :** Machine learning models can inadvertently perpetuate or amplify biases present in the training data, leading to unfair or discriminatory outcomes.

**Limitation :** Addressing biases and ensuring fairness in machine learning models requires careful consideration of data selection, feature engineering, and algorithmic design.

**Generalization to New Data:**

**Challenge :** Ensuring that machine learning models generalize well to unseen data is crucial for their effectiveness in real-world applications.

**Limitation :** Models may struggle to generalize to new data distributions, particularly if the distribution of the test data differs significantly from the training data.

## Introduction to Computer Vision

Computer vision is like giving computers the ability to see and understand the world, just like humans do with their eyes. It's a technology that allows computers to analyze and interpret visual information from images or videos. This could involve recognizing objects, understanding scenes, or even detecting movements. It's what enables things like facial recognition on your phone, self-driving cars to navigate, or helping doctors analyze medical images.

For example, let's say you have a security camera outside your house. With computer vision, the camera can detect if there's a person approaching your door, a car passing by, or even if a package has been left on your doorstep. It can analyze these visual inputs and alert you accordingly, enhancing security and convenience.

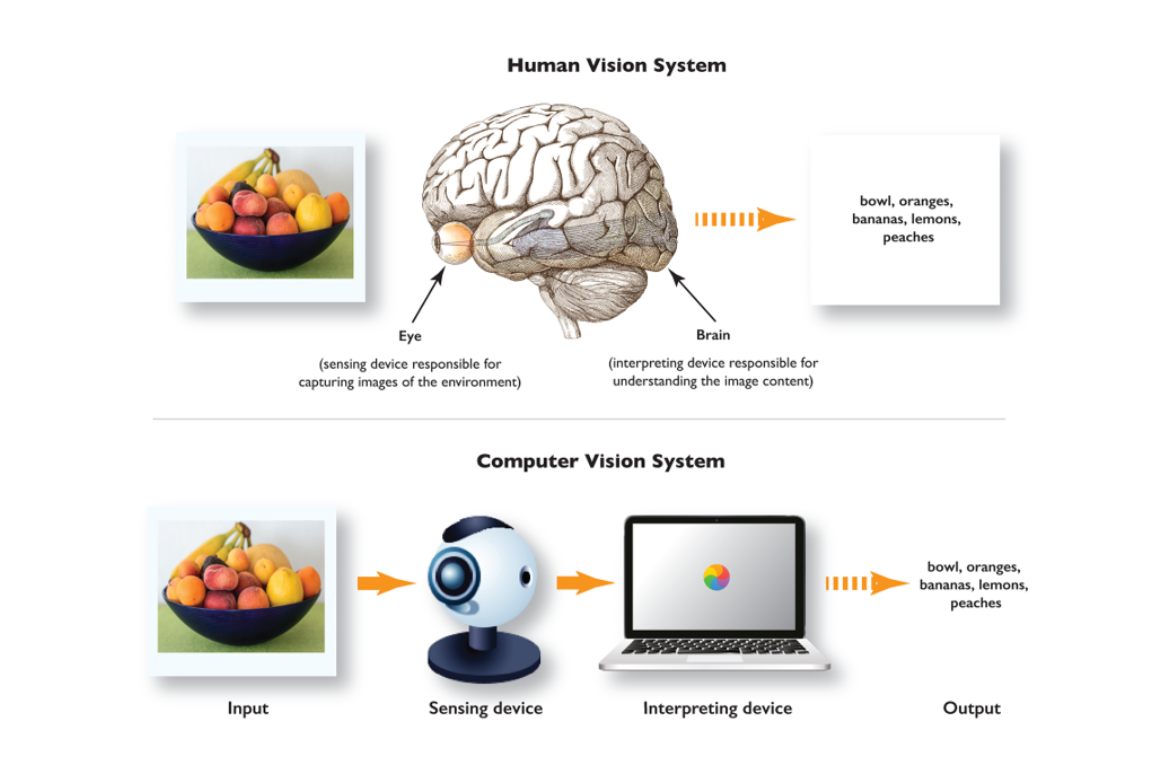


Figure 1.6 Human Vision vs Computer Vision

### How Computer Vision Works?

Generally, computer vision works in three basic steps:

Step 1: Acquiring the image/video from a camera,

Step 2: Processing the image, and

Step 3:Understanding the image.

**A Practical Example of Computer Vision**

Computer vision machine learning requires a massive amount of data to train a deep learning algorithm or machine learning alogorithms that can accurately recognize images. For example, to train a computer to recognize a helmet, it needs to be fed large quantities of helmet images with people wearing helmets in different scenes to learn the characteristics of a helmet. Next, the trained algorithm can be applied to newly generated images, for example, videos of surveillance cameras, to recognize a helmet.

This is, for example, used in computer vision applications for equipment inspection to reduce accidents in constructing or manufacturing

Modern computer vision systems combine image processing with machine learning and deep learning techniques. Hence, developers combine different software (etc., OpenCV or OpenVINO) and AI algorithms to create a multi-step process, a computer vision pipeline.

The organization and setup of a computer vision system vary based on the application and use case. However, all computer vision systems contain the same typical functions:

* **Step 1:** Image acquisition. The digital image of a camera or image sensor provides the image data or video. Technically, any 2D or 3D camera or sensor can be used to provide image frames.
* **Step 2:** Pre-processing. The raw image input of cameras needs to be preprocessed to optimize the performance of the subsequent computer vision tasks. Pre-processing includes noise reduction, contrast enhancement, re-scaling, or image cropping.
* **Step 3:** Computer vision algorithm. The image processing algorithm, most popularly a machine learning or deep learning model (DL model), performs image recognition, object detection, image segmentation, and classification on every image or video frame.
* **Step 4:** Automation logic. The AI algorithm output information needs to be processed with conditional rules based on the use case. This part performs automation based on information gained from the computer vision task. For example, pass or fail for automatic inspection applications, match or no-match in recognition systems, flag for human review in insurance, surveillance and security, military, or medical recognition applications.

### Applications of computer vision.webpApplications of Computer Vision

Figure 1.7 Computer Vision Application

**Facial recognition:** Utilizes computer vision to identify individuals by analyzing unique facial features, enabling applications like unlocking smartphones or enhancing security systems.

**Self-driving vehicles:** Relies on computer vision to process real-time images and create 3D maps, allowing autonomous vehicles to navigate roads, detect obstacles, and ensure passenger safety.

**Healthcare:** Utilizes computer vision for medical imaging analysis, aiding in accurate diagnoses and treatment planning for various medical conditions.

**Augmented reality (AR):** Integrates computer vision to overlay digital information onto real-world environments, enhancing experiences like measuring objects using a smartphone camera.

**Super-resolution imaging (SR):** Employs computer vision techniques to enhance image resolution, utilizing algorithms like EDSR and ESPCNN to generate higher-quality images from low-resolution inputs.

**Optical character recognition (OCR):** Utilizes computer vision to extract text from images and scanned documents, enabling tasks like digitizing text, passport scanning, and data analysis.

### Fundamental of Image Processing

**Image Representation**

**Pixels:** Digital images are composed of tiny picture elements called pixels. Each pixel represents a single color value, typically expressed as a combination of red, green, and blue (RGB) intensities. For example, a pixel may have values like (255, 0, 0) representing pure red, (0, 255, 0) representing pure green, and (0, 0, 255) representing pure blue.

**Resolution:** Image resolution refers to the number of pixels in an image, affecting its clarity and detail. Higher resolutions result in sharper images with more detail, while lower resolutions may appear pixelated or blurry.

**Color Spaces:** Common color spaces like RGB (Red, Green, Blue) and grayscale are used to represent colors in digital images. In the RGB color space, colors are represented as combinations of red, green, and blue intensities. Grayscale images contain only one channel representing the intensity of brightness, ranging from black to white.

**Image Filtering and Enhancement**

**Filtering:** Filters are applied to images to enhance or modify specific features. For example, blurring filters reduce noise and smooth out details, sharpening filters enhance edges and fine details, and smoothing filters remove imperfections. Filters work by modifying the pixel values in the image according to predefined mathematical operations.

**Histogram Equalization:** Histogram equalization is a technique used to improve contrast in images by redistributing pixel intensity values. It works by stretching the histogram of pixel intensities to cover the entire dynamic range, resulting in enhanced contrast and improved visibility of details.

**Feature Detection and Extraction**

**Edge Detection:** Edge detection algorithms identify abrupt changes in pixel intensity, highlighting object boundaries and contours in images. These algorithms search for areas where pixel intensity changes sharply, indicating the presence of edges or boundaries between objects in the image.

**Corner Detection:** Corner detection is a method used to identify key points where pixel intensities change in multiple directions. These key points, known as corners or interest points, are useful for object recognition and tracking tasks. Corner detection algorithms

search for regions where the gradient of pixel intensities is high in multiple directions, indicating the presence of corners or junctions in the image.

### Computer Vision Techinques

**Image Classification**

Image classification is the simplest technique of Computer Vision. The main aim of image classification is to classify the image into one or more different categories. Image classifier basically takes an image as input and tells about different objects present in that image, such as a person, dog, tree, etc. However, it would not give you other more information about the image data, such as how many persons are there, tree colour, item positions, etc., and for this, we need to go for any other CV technique.

Image classification is basically of two types, Binary classification and multi-class classification. As the name suggests, binary image classification looks for a single class in the given image and provides results based on if the image has that object or not. For example, we can achieve superhuman performance in detecting skin cancer in humans by training an AI system on both images that have skin cancer and images that do not have skin cancer.

**Object Detection**

Object detection is another popular technique of computer vision that can be performed after Image classification or which uses image classification to detect the objects in visual data. It is basically used to recognize the objects within the boundary boxes and find the class of the objects in the image. Object detection makes use of deep learning and machine learning technology to generate useful results.

As human beings, whenever we see a visual or look at an image or video, we can immediately recognize and even locate the objects within a moment. So, the aim of object detection is to replicate the same human intelligence into machines to identify and locate the objects.

Object detection has several applications, including object tracking, retrieval, video surveillance, image captioning, etc.

A variety of techniques can be used to perform object detection, which includes R-CNN, YOLO v2, etc.

**Semantic Segmentation**

Semantic Segmentation is not only about detecting the classes in an image as image classification. Instead, it classifies each pixel of an image to specify what objects it has. It tries to determine the role of each pixel in the image. It basically classifies pixelS in a particular category without differentiating the object instances. Or we can say it classifies similar objects as a single class from the pixel levels. For example, if an image contains two dogs, then semantic segmentation will put both the dogs under the same label. It tries to understand the role of each pixel in an image.

singly entities, whereas instance segmentation classified all the persons as different by considering colours also.

### Challenges and Future Directions in Computer Vision

**Challenges in Computer Vision**

While computer vision has made significant advancements, several challenges still exist, and addressing them is crucial for further progress. Here are some key challenges in computer vision

**Ambiguity and Variability**

Images and videos can exhibit variations in lighting, viewpoint, scale, occlusions, deformations, and background clutter. These factors introduce ambiguity and make object recognition, tracking, and scene understanding challenging tasks.

**Limited Training Data**

Developing accurate and robust computer vision models often requires large amounts of labeled training data. Obtaining annotated datasets for specific tasks can be time-consuming, expensive, and may suffer from biases or domain-specific limitations.

Computational Complexity

Many computer vision algorithms are computationally demanding, requiring substantial processing power and memory resources. Real-time processing or deployment on resource-constrained devices can pose significant challenges.

**Future Directions in Computer Vision**

As computer vision continues to evolve, several research directions and trends are shaping its future. Here are some key areas of focus

**Deep Learning and Neural Architectures**

Deep learning, particularly convolutional neural networks (CNNs), has revolutionized computer vision. Future research aims to develop more efficient architectures, explore self-supervised learning, and enhance model interpretability to improve performance and address limitations.

**Generative Models and Image Synthesis**

Advancements in generative models, such as Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), are enabling tasks like image synthesis, data augmentation, and unsupervised representation learning. Further research in generative models will lead to improved realism and control over synthesized images.

**Explainability and Interpretability**

As deep learning models become more complex, there is a growing need for interpretability and explainability. Research focuses on developing methods to understand model decisions, attribute importance to features, and provide human-understandable explanations for computer vision algorithms.

## Motivation

Imagine driving into a busy parking lot, circling around endlessly just to find a vacant spot. It's frustrating, time-consuming, and adds unnecessary stress to your day. Now, picture a world where you could simply glance at your phone and instantly know where the empty parking spaces are. That's the motivation behind this project.

In today's world, where cities are getting more crowded and parking spaces are becoming scarcer, this system aims to make our lives a bit easier. By using smart cameras and clever algorithms, it helps us efficiently manage parking lots by identifying available spots in real-time. This not only saves us time but also reduces traffic congestion and pollution caused by endless circling in search of parking.

## Problem Statement

In today's crowded world, finding parking can be a major hassle. It's frustrating, time-consuming, and often leads to unnecessary stress. As cities grow, the demand for parking spaces increases, making it even harder to find a spot. This project aims to tackle this problem by developing a Parking Space Detection System. The goal is to use technology to make it easier for drivers to find parking quickly and efficiently. By using cameras and smart algorithms, the system will be able to identify empty parking spaces in real-time, helping drivers navigate to available spots without the frustration of endless searching. This project seeks to improve the parking experience for everyone, reducing congestion, saving time, and making our cities more livable

## Scope of The Project

We're building a smart system that can "see" and understand parking spaces using cameras and computer intelligence. By training a special kind of computer brain called a machine learning model, we're teaching it to recognize when a parking spot is empty or occupied. This means that drivers will be able to know in real-time where they can park without having to search endlessly. It's like having a helpful assistant guiding you to an available spot, making parking a breeze and saving you time and frustration. So, the scope of this project is to make parking easier and more convenient for everyone using the power of technology.

# Literature Survey

The purpose of this literature survey is to explore and analyze various technologies and methodologies employed in car parking systems. By reviewing existing research and developments in the field, we aim to gain insights into the diverse range of solutions available for addressing parking-related issues in urban areas.

**1."IOT Based Car Parking"**

Patil, P. S., Almel, N. D., Padaganur, S. K., Ayyangoudar, B., Gokak, M. R., & Mirajkar, K. (2020).

* This paper presents an IoT-based car parking system implemented using sensors and NodeMCU.
* It focuses on utilizing ultrasonic sensors for detecting parking space occupancy and transmitting data to a centralized system.
* The system aims to improve parking management and reduce congestion in urban areas. The study provides insights into the integration of IoT technologies for efficient parking solutions.

**2."Design and Development of Smart Car Parking System using Internet of Things"**

Rashid, A., Salahuddin, M. A., Islam, M. S., & Islam, M. S. (2018).

* This paper discusses the design and development of a smart car parking system using IoT technologies.
* It explores the implementation of sensors and microcontrollers to monitor parking space occupancy in real-time.
* The system aims to provide users with information about available parking spaces, thereby improving parking efficiency and reducing search time. The study contributes to the advancement of IoT-based parking solutions.

**3."Automated Car Parking System using Internet of Things"**

Shaikh, S. S., & Deshpande, M. (2019).

* This paper presents an automated car parking system leveraging IoT technologies.
* It focuses on the utilization of sensors and microcontrollers for detecting parking space occupancy and managing parking operations.
* The system aims to optimize parking space utilization and enhance user convenience. The study demonstrates the effectiveness of IoT-based solutions in addressing parking challenges and improving urban mobility.

This survey will cover a wide spectrum of technologies and approaches, including Internet of Things (IoT)-based solutions, machine learning algorithms, sensor technologies, and other innovative techniques. By examining the strengths, limitations, and applications of each technology, we hope to identify promising avenues for improving parking management and enhancing the overall urban transportation infrastructure.

# System Analysis

## Existing System

The project module proposes an IoT-based solution to the difficulty of finding parking spaces in metropolitan areas. It integrates Arduino Uno, Ultrasonic sensor, LCD display, servo motors, push buttons, and an Android application to detect parking space availability. Ultrasonic sensors measure distance and motion, transferring data to Arduino, then to the Android app via Bluetooth. LED indicators in the app offer real-time parking status, providing drivers with a streamlined experience and reducing traffic congestion.

**IOT-BASED CAR PARKING**

**Introduction:**

The project presents an IoT-based solution to the daunting task of finding parking in metropolitan areas. By leveraging an Arduino-based device integrated with sensors and a telnet app, it aims to provide drivers with real-time updates on available parking spaces via their smartphones. This innovative approach eliminates the hassle of manual searching and enhances efficiency. The project's main objective is to develop a demonstration unit with five parking spaces, emphasizing the IoT aspect to facilitate seamless communication between the hardware components and the mobile application, ultimately aiding drivers in navigating urban parking challenges.

### Methodology

1. **Entry Detection:** When a customer enters the parking garage, the system begins by detecting the entry of the vehicle onto the platform.

2. **IR Sensor Operation:** When a vehicle is parked in front of an IR sensor, it emits IR rays. These rays are then reflected and detected by a photodiode. The sensor sends this information to a controller.

3. **Controller Response:** The controller processes the information received from the IR sensor. If a slot is occupied, it activates the corresponding RED LED. If the slot is available, it activates the GREEN LED.

4. **LED Indicators:** LED indicators are installed for each parking slot. When a slot is occupied, a RED LED lights up, indicating its unavailability. Conversely, when a slot is vacant, a GREEN LED illuminates.

5. **Driver Interface - Stage 1:** Upon entering the parking area, the driver can access an Android app to check the availability of parking slots. Green LEDs indicate vacant slots, while red LEDs indicate occupied slots.

6. **Driver Interface - Stage 2:** Inside the parking area, an LCD display is utilized to provide real-time information about the availability of parking slots. This allows drivers to make informed decisions about where to park.

7. **TELNET Application:** The system offers an additional interface for users through a TELNET application. This provides users with status updates on available parking slots, enhancing convenience and efficiency.

By following these steps, the methodology ensures that drivers can easily locate available parking spaces, reducing congestion and improving the overall parking experience in metropolitan areas.

### Drawbacks

**Limited Accuracy:** IR sensors may have constraints in accurately detecting parked vehicles, leading to potential false positives or negatives.

**Dependency on Hardware:** The system heavily relies on physical components like IR sensors and microcontrollers, making it susceptible to malfunctions or damage.

**Limited Scalability:** Scaling the system to cover larger parking areas with numerous slots may pose challenges in terms of cost and complexity.

**Static Configuration:** System settings, such as IR sensor thresholds, remain fixed and may not adapt well to changing environmental conditions or parking dynamics.

## Proposed System

Our current proposed system utilizes computer vision and machine learning techniques for parking spot detection. Specifically, we employ computer vision algorithms to process video footage from a parking lot and extract relevant features from the images. Then, machine learning algorithms, such as Support Vector Machines (SVM), are trained to classify parking spots as either 'empty' or 'not empty' based on the extracted features. This approach enables real-time monitoring of parking spot availability in the parking lot.

### How Model Trained?

**1.Data Preparation**

* Images of parking spots are collected and organized into two categories: 'empty' and 'not\_empty'.
* Each image is loaded and resized to a fixed size of 15x15 pixels. Resizing ensures uniformity in the dimensions of all images, which is essential for training a machine learning model.
* The pixel values of each resized image are flattened into a one-dimensional array. This process converts each image into a feature vector, making it suitable for input to the machine learning model.
* A black rectangular object with white text

  Description automatically generatedCorresponding labels (0 for 'empty' and 1 for 'not\_empty') are assigned to each image, indicating the class to which it belongs.

Figure 3.1 Empty DatasetA group of rectangular objects with text

Description automatically generated with medium confidence

Figure 3.2 Not\_Empty Dataset

**2. Train/Test Split**

* The dataset is divided into training and testing sets to evaluate the performance of the trained model.
* The train\_test\_split function randomly splits the data into training and testing sets, with 80% of the data used for training and 20% for testing.
* The stratify parameter ensures that the class distribution in the training and testing sets remains similar to the original dataset, preventing bias in the model evaluation.

**3. Model Selection and Training:**

* A Support Vector Classifier (SVC) is chosen as the classification algorithm for its effectiveness in binary classification tasks.
* Hyperparameters of the SVC, namely gamma and C, are tuned using grid search with cross-validation (GridSearchCV). Grid search exhaustively searches through a specified parameter grid to find the optimal combination of hyperparameters that yields the best model performance.

The grid search object is fit to the training data to train multiple SVM models with different hyperparameter settings and select the one with the highest performance.

**4. Model Evaluation:**

* The performance of the trained model is evaluated on the testing set to assess its generalization ability.
* Predictions are made on the testing set using the best estimator obtained from grid search.
* The accuracy of the predictions is calculated by comparing them with the ground truth labels from the testing set. The accuracy score indicates the proportion of correctly classified samples in the testing set, providing a measure of the model's effectiveness in distinguishing between 'empty' and 'not\_empty' parking spots.

A diagram of classification metrics

Description automatically generatedFigure 3.3 Classification Metrics

**5. Model Serialization:**

* The best performing SVM model is serialized using the pickle module and saved to a file named 'model. p'. Serialization allows the trained model to be stored on disk and reused later for making predictions on new data without needing to retrain the model.

### How Model Works?

* **Image and Video Upload:** Users upload an image containing a mask defining the parking spots and a video showing the parking area.
* **Connected Component Analysis:** The uploaded image is read and converted to grayscale. Then, the cv2.connectedComponentsWithStats() function is used to identify connected components in the binary mask image. Connected components represent different regions of the image that are contiguous.
* **Bounding Box Extraction:** The function get\_parking\_spots\_boxes() extracts bounding boxes around the identified connected components. These bounding boxes represent the regions where parking spots are located.
* **Frame Processing:** The uploaded video is read frame by frame. For each frame, the algorithm calculates the difference between the current frame and the previous frame. This difference is used to determine if there are any changes in the parking spots, such as cars entering or leaving.
* **Spot Status Determination:** For each identified parking spot, a crop of the frame is taken corresponding to the bounding box of the spot. The function empty\_or\_not() is then used to determine whether the spot is empty or occupied based on the contents of the cropped image.
* **Visualization:** The status of each parking spot is visualized on the frame. If a spot is empty, it is outlined in green; otherwise, it is outlined in red. Additionally, the number of available spots is displayed on the frame.

# Implementation

## Software Requirements

In this study we are using python as the coding language and the implementation can be done in any of its environments such as Jupyter Notebook, Google Colab…etc. We don’t even require any kind of download & installation of tool that are available online. We can access The Google Colab through our google accounts. For faster Execution prefer Jupyter Notebook as it doesn’t require uploading of files all the time.

**Python**

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's "benevolent dictator for life", a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker.

In January 2019, active Python core developers elected a five-member Steering Council to lead the project. As of November 2022, Python 3.11.0 is the current stable release. Notable changes from 3.10 include increased program execution speed and improved error reporting.

## Libraries and APIs Used

For our current working/developing/proposed system we require some libraries as well as API’s (Application Programming Interface) such as :

* Flask
* OpenCV-Python
* Scikit-learn
* Numpy
* Pillow
* Scikit-image
* Matplotlib

**Flask :**

Flask is a lightweight web application framework for Python. It's designed to make it easy to build web applications quickly and with minimal boilerplate code. Flask is known for its simplicity and flexibility, making it a popular choice for developing web applications of various scales, from small projects to larger, more complex ones.

Here are some key features of Flask :

* **Routing** : Flask allows you to define routes that map URLs to Python functions. This makes it easy to handle different HTTP requests (GET, POST, etc.) and serve dynamic content based on user input.
* **Template Engine :** Flask comes with a built-in template engine called Jinja2, which allows you to generate HTML dynamically. Templates can include placeholders, loops, and conditionals, making it easy to generate dynamic web pages.
* **Development Server :** Flask comes with a built-in development server that makes it easy to run and test your applications locally during development. The development server automatically reloads your application when code changes

**OpenCV-Python:**

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. It provides a wide range of functionalities for real-time computer vision tasks, such as image processing, object detection, video analysis, and machine learning.

In our project, OpenCV (specifically, the cv2 module) is used extensively for various tasks:

* **Video Processing:** OpenCV’s video capture (cv2.VideoCapture) functionality is utilized to capture frames from a video source (e.g., a video file or camera feed).The captured frames are processed to detect parking spaces' occupancy status and visualize the results.
* **Image Processing:** OpenCV provides numerous image processing functions for tasks like image resizing, filtering, and edge detection. In the project, OpenCV is used to resize images to a fixed size for model training and processing parking space regions in video frames.
* **Connected Component Analysis:** Connected component analysis is a technique used to label connected regions in binary images. OpenCV’s cv2.connectedComponentsWithStats function is employed to identify connected components in a binary mask image representing parking space regions.
* **Drawing on Images:** OpenCV allows drawing shapes (e.g., rectangles) and text on images, which is crucial for visualizing the results of parking space occupancy detection. Detected parking spaces and their occupancy status are highlighted by drawing rectangles and displaying text on video frames using OpenCV's drawing functions.

**Scikit-learn :**

Scikit-learn is a popular open-source machine learning library for Python. It provides a wide range of tools for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, and model selection. Scikit-learn is built on top of other Python libraries such as NumPy, SciPy, and matplotlib, making it easy to integrate into Python-based machine learning workflows.

In our project, scikit-learn is used primarily for training a machine learning model to classify parking spaces as either empty or occupied. Here's how scikit-learn is used:

* **Data Preparation:** Scikit-lean’s train\_test\_split function is utilized to split the dataset into training and testing sets. This allows the model to be trained on one subset of the data and evaluated on another to assess its performance.
* **Model Training:** The project employs scikit-lean’s Support Vector Machine (SVM) classifier (sklearn.svm.SVC) to build a model for classifying parking spaces.
* Grid search (sklearn.model\_selection.GridSearchCV) is used to perform hyperparameter tuning, optimizing the SVM model's performance by searching for the best combination of hyperparameters (e.g., gamma and C values) using cross-validation.
* **Model Evaluation:** Scikit-lean’s accuracy score function from sklearn.metrics is used to evaluate the model's performance on the test set. It computes the accuracy of the model by comparing the predicted labels to the true labels.
* **Model Persistence:** The trained SVM model is serialized using Python's pickle module (pickle.dump) to save it to a file (model.p). This allows the model to be loaded and used later without retraining, enabling efficient deployment in the parking space detection system.

**Numpy:**

NumPy is a fundamental package for numerical computing in Python. It provides support for multidimensional arrays, along with a collection of mathematical functions to operate on these arrays efficiently. NumPy is widely used in scientific computing, data analysis, and machine learning tasks due to its speed and versatility.

In our project, NumPy is used primarily for handling image data and performing array operations. Here's how NumPy is utilized:

* **Image Data Representation:** NumPy arrays are used to represent image data. Images are read using functions from the skimage.io module and converted into NumPy arrays. These arrays are multidimensional, typically with dimensions representing the image's height, width, and color channels.
* **Array Operations:** Various array operations are performed on the image data using NumPy. For example, the numpy.resize function is used to resize images to a specified shape, ensuring consistency in input dimensions for the machine learning model.
* The numpy.ndarray.flatten method is applied to flatten the multidimensional arrays representing images into one-dimensional arrays. This flattening process is often necessary when preparing data for machine learning algorithms that expect one-dimensional input.
* **Data Preparation:** NumPy arrays are employed to store the image data and labels during data preparation. Images are flattened and stored in NumPy arrays, while labels indicating whether a parking space is empty or occupied are stored as NumPy arrays of integers.
* **Mathematical Operations:** NumPy’s mathematical functions are used for various operations on image data or processed arrays. These operations may include calculating differences between images, computing statistics, or performing other transformations necessary for image processing or feature extraction.

**Pillow**

Pillow is a Python Imaging Library (PIL), which adds support for opening, manipulating, and saving many different image file formats. It is a widely used library for image processing tasks in Python due to its simplicity and versatility.

In our project, Pillow may be used for various image-related operations, such as loading images from files, resizing, and saving images. While the specific usage of Pillow is not evident from the code provided, it is common to use Pillow alongside libraries like OpenCV and scikit-image for image preprocessing and manipulation tasks. Here how it pillow is utilized

* **Loading Images:** Pillow can be used to load images from files into Python as PIL Image objects. These images can then be further processed or converted into NumPy arrays for analysis.
* **Resizing Images:** Pillow provides functions to resize images to specific dimensions. This functionality may be used to standardize the size of images before feeding them into machine learning models or for other preprocessing tasks.
* **Saving Images:** After processing or analysis, Pillow can save images to various file formats. This capability is useful for saving processed images or results for further examination or presentation.
* **Image Manipulation:** Pillow offers a range of image manipulation functions, such as cropping, rotating, and applying filters. These operations can be used for data augmentation, enhancing image quality, or extracting relevant features from images.

**Scikit-image :**

scikit-image is a Python library for image processing that is built on top of NumPy, SciPy, and matplotlib. It provides a collection of algorithms for image processing tasks such as filtering, segmentation, morphological operations, and feature extraction. Below is a breakdown of how scikit-image utilized:

* **Loading Images:** scikit-image can be used to load images from files into NumPy arrays, which can then be processed or analyzed further.
* **Preprocessing:** It offers various preprocessing techniques such as denoising, histogram equalization, and color space conversion. These techniques can be applied to enhance the quality of images or to prepare them for further analysis.
* **Feature Extraction:** scikit-image provides functions for extracting features from images, such as edges, corners, and texture descriptors. These features can be used for tasks like object detection, classification, or image segmentation.
* **Segmentation:** The library offers algorithms for image segmentation, which divides an image into regions or objects based on certain criteria. This can be useful for identifying parking spots or other regions of interest in the project.
* **Morphological Operations:** Morphological operations like erosion, dilation, opening, and closing are available in scikit-image. These operations are often used for cleaning up images or modifying shapes.
* **Visualization:** scikit-image includes tools for visualizing images and the results of various processing steps. This can aid in understanding and debugging the image processing pipeline.

**Matplotlib**

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It can be used to generate plots, histograms, bar charts, scatter plots, and more. In the provided project code, matplotlib is likely used for visualizing the following:

* **Displaying Images:** Matplotlib can be used to display images loaded using other libraries such as OpenCV or scikit-image. It provides functions like imshow() to show images in a window or notebook.
* **Plotting Graphs:** While not explicitly shown in the provided code snippets, matplotlib could be used to plot graphs or charts to visualize data related to the project. For example, it might be used to plot the accuracy of the trained model or to visualize the distribution of parking spot occupancy over time.
* **Annotating Images:** Matplotlib can also be used to annotate images with text, arrows, or shapes. This could be useful for adding additional information or annotations to the parking spot detection results displayed in the project.

# System Design

## Use Case Diagram.

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The Figure 5.1 shows the use case representation of the system.

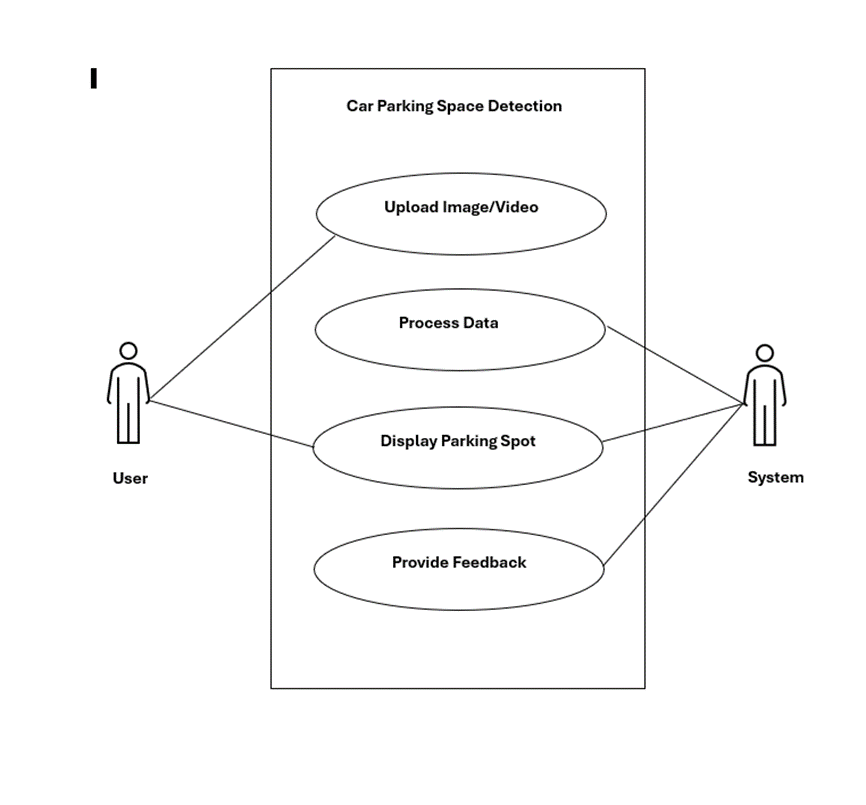


Figure 5.1 Usecase Diagram

## Class Diagram

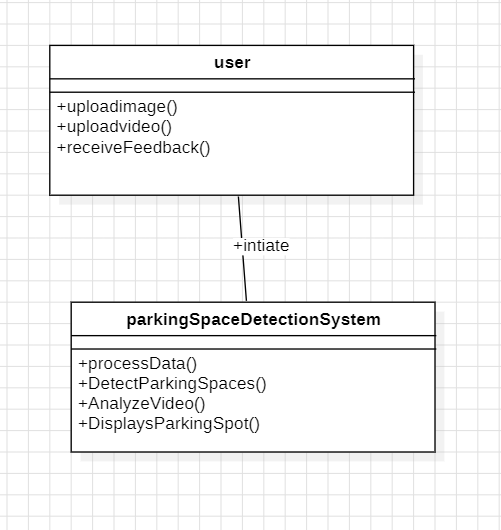
Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. The Figure 5.2 shows the class diagram representation of the system.

Figure 5.2 Class Diagram

## PlantUML diagramActivity Diagram.

Figure 5.3 Activity diagram.

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. In UML, an activity diagram provides a view of the behavior of a system by describing the sequence of actions in a process. The Figure 5.3 shows the activity diagram representation of the system.

## Sequence Diagram.

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. The Figure 5.4 shows the sequence diagram representation of the system. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Much like the class diagram, developers typically think sequence diagrams were meant exclusively for them.

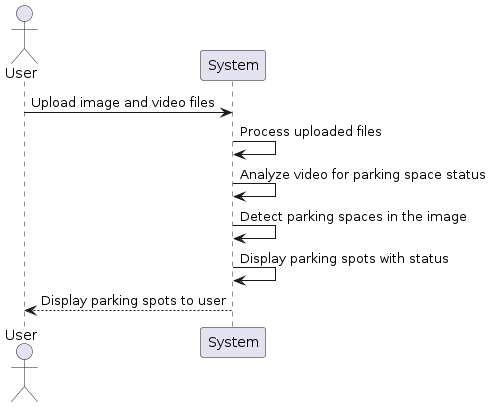


Figure 5.4 Sequence Diagram

## State Chart Diagram

A state diagram, also known as a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML). The Figure 5.5 shows the state chart diagram representation of the system.

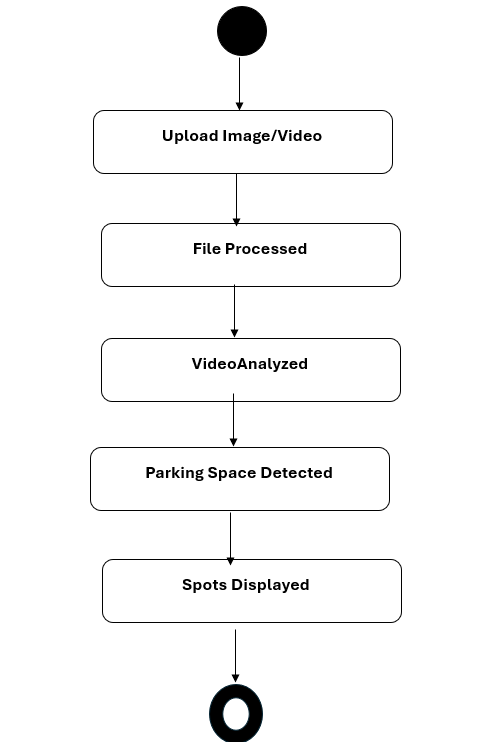


Figure 5.5 State Chart Diagram

# Result

Here are the steps involved in obtaining the result in a deep fake video detection project using LSTM:

* **Access Upload Page:** Visit the upload page (/upload) in Flask application.
* **Select Files:** Choose an image file containing the mask and a video file for detecting parking spots.
* **Upload Files:** Submit the selected files through the upload form.
* **File Processing:** The server processes the uploaded files, extracting information from the mask image and analyzing the video frames.
* **Spot Detection:** Using computer vision techniques, the application identifies parking spots within the video frames.
* **Display Results:** Processed video frames are displayed with rectangles indicating available in green rectangles and occupied in red rectangles around the parking spots.

Home page of the website

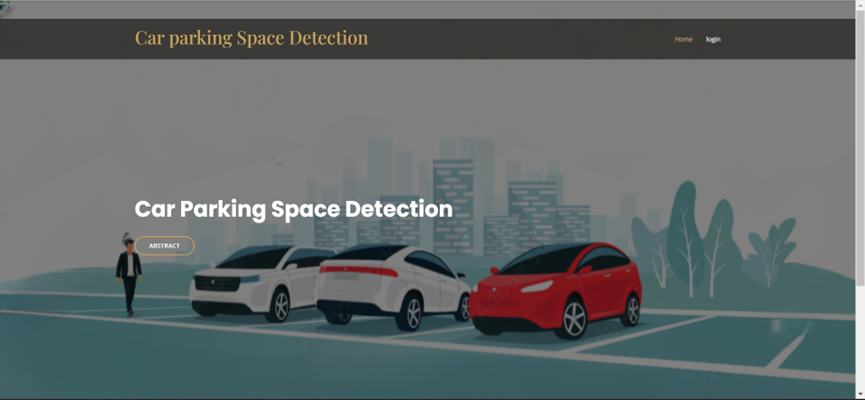


Figure 6.1 Home Page

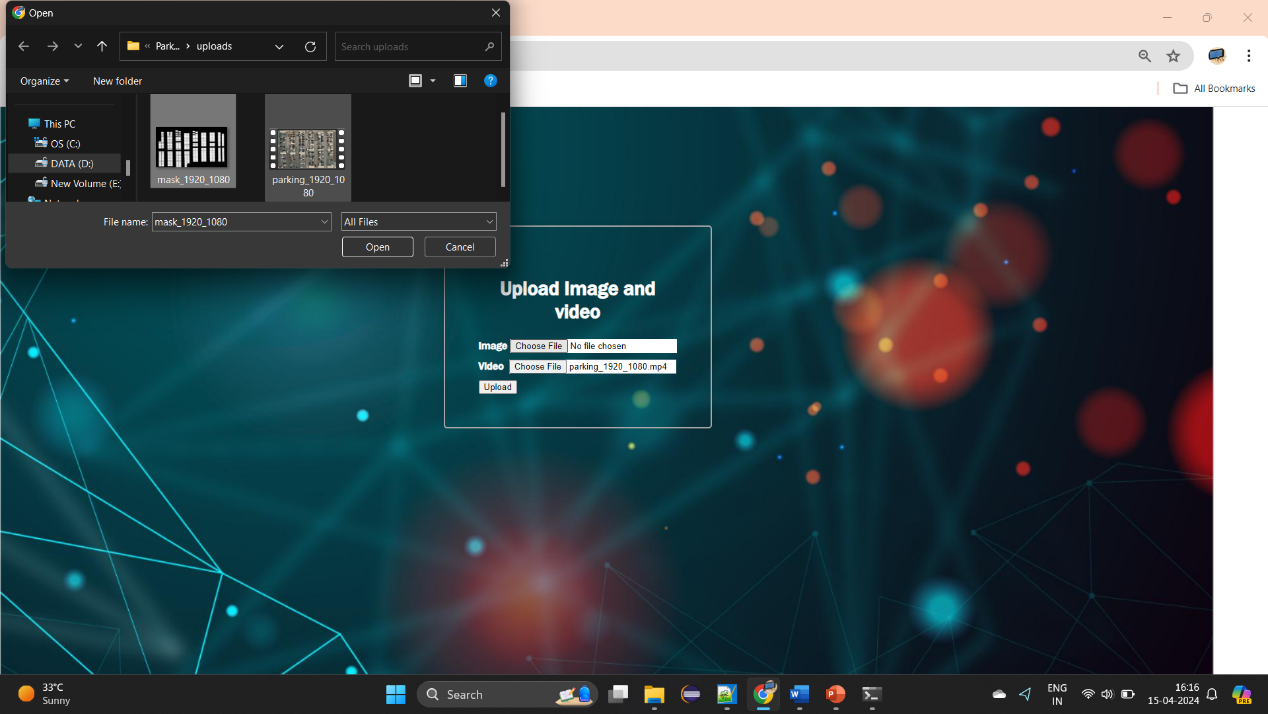
After Successful login into Website:

Figure 6.2 Upload Page

After Successful upload the mask image and video, model detect spots and display count, available spots [1].

Figure 6.3 Result Page

# Conclusion

In the era of urbanization and technological innovation, efficient management of parking spaces has become increasingly critical. The project represents a significant advancement in addressing this challenge through the integration of computer vision and web technologies. Leveraging Flask for web development and OpenCV for image processing, the system provides users with a seamless platform to upload images and videos of parking areas, enabling real-time detection and monitoring of parking spot availability.

One of the notable strengths of the system lies in its ability to overcome limitations commonly associated with traditional IoT-based car parking systems. While IoT systems rely heavily on sensor networks for data collection, the proposed solution harnesses the power of computer vision algorithms to analyze visual data from cameras. This approach offers several advantages, including greater flexibility in deployment, reduced infrastructure costs, and enhanced accuracy in detecting parking spot occupancy.

Furthermore, by utilizing machine learning algorithms and image analysis techniques, the system can adapt to diverse parking environments and dynamic conditions. This adaptability ensures robust performance across different locations and varying lighting conditions, thereby enhancing the reliability and effectiveness of parking space management.

In summary, the project marks a significant leap forward in modernizing parking space management. By leveraging computer vision and machine learning, it offers a flexible, cost-effective, and accurate solution for optimizing parking space utilization. As cities strive for smarter infrastructure, this innovative system promises to address the challenges of urban congestion and improve the overall efficiency of transportation networks.

# Bibliography

[1] Patil, P. S., Almel, N. D., Padaganur, S. K., Ayyangoudar, B., Gokak, M. R., & Mirajkar, K. (2020). "IOT Based Car Parking." IEEE DOI:10.1109/B-HTC50970.2020.9298002. Conference: 2020 IEEE Bangalore Humanitarian Technology Conference (B-HTC).

[2] Rashid, A., Salahuddin, M. A., Islam, M. S., & Islam, M. S. (2018). "Design and Development of Smart Car Parking System using Internet of Things." 2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2).

[3] Shaikh, S. S., & Deshpande, M. (2019). "Automated Car Parking System using Internet of Things." 2019 International Conference on Sustainable Computing and Advanced Artificial Intelligence (ICSCAA). IEEE.

[4] Maheshwari, P., & Tyagi, S. (2019). "Smart Car Parking System using Internet of Things." 2019 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE). IEEE.

[5] Zhang, J., Ma, L., Jiang, H., Cao, S., Wang, H., & Wei, H. (2019). "A Vehicle Detection Method Based on Deep Learning in Parking Lot." 2019 IEEE International Conference on Image Processing (ICIP). IEEE.

[6] Hsu, C. L., & Shih, C. S. (2019). "A Fast and Low-Cost Detection Method of Empty Parking Spaces Based on Convolutional Neural Networks." 2019 IEEE 5th International Conference on Computer and Communications (ICCC). IEEE.

[7] Bhuvaneswari, K., & Krishnaraj, N. (2020). "Real-Time Car Parking Slot Detection System Using OpenCV." 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS). IEEE.

[8] Guo, C., Yao, Y., & Wang, J. (2020). "An Object Detection Method of Parking Lot Based on Single Shot MultiBox Detector." 2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA). IEEE.

[9] Hu, Y., Zhu, Y., Chen, J., & Shang, Y. (2021). "Research on Car Parking Space Detection Algorithm Based on Deep Learning." 2021 International Conference on Artificial Intelligence and Big Data (ICAIBD). IEEE.

[10] Chen, J., & Wu, L. (2020). "Vehicle Detection and Classification for Intelligent Transportation System Based on Deep Learning." 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). IEEE.

[11] Zhang, H., Liu, L., & Zhang, M. (2019). "Vehicle Detection and Recognition Based on Deep Learning in Intelligent Traffic System." 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC). IEEE.