**A Project Report on**

**CAR PARKING SPACE DETECTION**

submitted in partial fulfillment for the award of

**Bachelor of Technology**

in

**Computer Science and Engineering**

by

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

**A blue logo with a white background

Description automatically generated with medium confidenceP Vyshnavi (Y20ACS537) Sk Asifa (Y20ACS561)**

Under the guidance of

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

Assistant Professor

Department of Computer Science and Engineering

**Bapatla Engineering College**

(Autonomous)

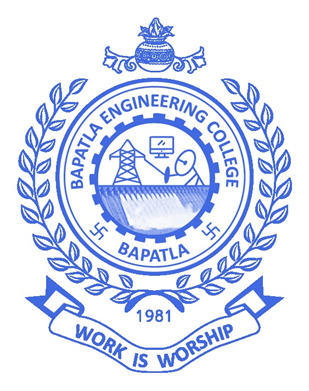
(Affiliated to Acharya Nagarjuna University)

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

**2023-2024**

**Department of**

**Computer Science and Engineering**

****

**CERTIFICATE**

This is to certify that the project report entitled **Car Parking Space Detection** that is being submitted by P Lakshmi Sahithi (Y20ACS533), P Hanumantha 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 and 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 Assoc. 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 the 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 valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr. M. Rajesh Babu**, Associate Professor & Head of the Department 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,** Professor, Department 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)**

**Table of Contents**

[List of Figures vii](#_Toc165030874)

[Abstract viii](#_Toc165030875)

[1 Introduction 1](#_Toc165030876)

[1.1 Machine Learning 1](#_Toc165030877)

[1.2 Supervised Machine Learning 2](#_Toc165030878)

[1.2.1 Types of Supervised Machine Learning 3](#_Toc165030879)

[1.3 Unsupervised Machine Learning 4](#_Toc165030880)

[1.3.1 Types of Unsupervised Machine Learning 5](#_Toc165030881)

[1.4 Reinforcement Machine Learning 6](#_Toc165030882)

[1.5 Machine Learning vs Traditional Programming 7](#_Toc165030883)

[1.6 How Does Machine Learning Work? 8](#_Toc165030884)

[1.6.1 Challenges and Limitations of Machine Learning 10](#_Toc165030885)

[1.7 Introduction to Computer Vision 12](#_Toc165030886)

[1.7.1 How Computer Vision Works? 12](#_Toc165030887)

[1.7.2 Applications of Computer Vision 14](#_Toc165030888)

[1.7.3 Fundamental of Image Processing 15](#_Toc165030889)

[1.7.4 Computer Vision Techniques 17](#_Toc165030890)

[1.7.5 Challenges and Future Directions in Computer Vision 19](#_Toc165030891)

[1.8 Motivation 21](#_Toc165030892)

[1.9 Problem Statement 21](#_Toc165030893)

[1.10 Scope of The Project 22](#_Toc165030894)

[2 Literature Survey 23](#_Toc165030895)

[3 System Analysis 25](#_Toc165030896)

[3.1 Existing System 25](#_Toc165030897)

[3.1.1 Methodology 26](#_Toc165030898)

[3.1.2 Drawbacks 27](#_Toc165030899)

[3.2 Proposed System 27](#_Toc165030900)

[3.2.1 How Model Is Trained? 28](#_Toc165030901)

[3.2.2 How Model Works? 30](#_Toc165030902)

[4 Implementation 32](#_Toc165030903)

[4.1 Software Requirements 32](#_Toc165030904)

[4.1.1 Python 32](#_Toc165030905)

[4.2 Libraries and APIs Used 33](#_Toc165030906)

[4.2.1 Flask 33](#_Toc165030907)

[4.2.2 OpenCV-Python 34](#_Toc165030908)

[4.2.3 Scikit-learn 35](#_Toc165030909)

[4.2.4 Numpy 36](#_Toc165030910)

[4.2.5 Pillow 37](#_Toc165030911)

[4.2.6 Scikit-image 38](#_Toc165030912)

[4.2.7 Matplotlib 39](#_Toc165030913)

[4.3 GitHub Repository Having Project Code 40](#_Toc165030914)

[5 System Design 41](#_Toc165030915)

[5.1 Usecase Diagram 42](#_Toc165030916)

[5.2 Class Diagram 43](#_Toc165030917)

[5.3 Activity Diagram 44](#_Toc165030918)

[5.4 Sequence Diagram 45](#_Toc165030919)

[5.5 State Chart Diagram 46](#_Toc165030920)

[6 Results 47](#_Toc165030921)

[7 Conclusions 49](#_Toc165030922)

[8 Bibliography 50](#_Toc165030923)

**List of Figures**

[Figure 1.1 Traditional Programming 7](#_Toc165030756)

[Figure 1.2 Machine Learning 7](#_Toc165030757)

[Figure 1.3 Learning Phase 8](#_Toc165030758)

[Figure 1.4 Inference from Model 9](#_Toc165030759)

[Figure 1.5 Human Vision vs Computer Vision 12](#_Toc165030760)

[Figure 1.6 Computer Vision Application 14](#_Toc165030761)

[Figure 3.1 Empty Dataset 28](#_Toc165030762)

[Figure 3.2 Not\_empty Dataset 29](#_Toc165030763)

[Figure 3.3 Classification Metrics 30](#_Toc165030764)

[Figure 5.1 Usecase Diagram 42](#_Toc165030765)

[Figure 5.2 Class Diagram 43](#_Toc165030766)

[Figure 5.3 Activity Diagram 44](#_Toc165030767)

[Figure 5.4 Sequence Diagram 45](#_Toc165030768)

[Figure 5.5 State Chart Diagram 46](#_Toc165030769)

[Figure 6.1 Home Page 47](#_Toc165030770)

[Figure 6.2 Upload Page 48](#_Toc165030771)

[Figure 6.3 Result Page 48](#_Toc165030772)

**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.

**Keywords:** SVM, Computer Vision, occupancy prediction, spot classification.

# Introduction

Our project focuses on optimizing urban parking management through the implementation of a Support Vector Machine (SVM) classifier. Leveraging computer vision and machine learning principles, this classifier enables real-time detection of parking spaces by analysing video feeds from parking areas. With a clear emphasis on simplicity and effectiveness, our solution aims to address the pressing need for efficient parking space management in densely populated urban environments.

By seamlessly integrating fundamental computer vision and machine learning techniques, our project endeavours to revolutionize urban infrastructure. Through the deployment of SVM-based classification algorithms, parking administrators gain access to valuable insights regarding parking spot occupancy in real time. This empowers them to make informed decisions regarding resource allocation and parking space utilization, ultimately contributing to smarter, more efficient, and sustainable 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.

## **Supervised Machine Learning**

Supervised Machine Learning utilizes labeled data for training, where each data point has a known output. The model learns from this labeled data to understand the relationship between input features and their associated outputs. By generalizing from the training data, the model can make accurate predictions on new, unseen data based on learned patterns. This approach is fundamental in various applications, such as classification, regression, and recommendation systems, where predicting specific outcomes from input features is essential. Here is how a supervised algorithm works:

1. You feed it an example input, then the associated output.
2. You repeat the above step many times. Eventually, the algorithm picks up a pattern between the inputs and outputs.
3. Now, you can feed it a new input, and it will predict the output for you.

**Training Process:** Supervised learning requires external guidance during training, utilizing labeled data to guide the model's learning process. Each input is associated with a corresponding output or target value, facilitating the mapping of inputs to outputs. Through iterative adjustments, the model minimizes the discrepancy between predicted and actual values, optimizing its performance. This approach is ideal for tasks where correct outputs are available during training, enabling the model to learn from labeled examples and make accurate predictions on unseen data.

**Applications of Supervised Learning**

1. Weather Prediction.
2. Sales Forecast.
3. Stock Price Analysis.

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

1. Linear Regression.
2. Logistic Regression.
3. Support Vector Machines.
4. K Nearest Neighbors.

**Eg.:** Suppose your friend gives you 1 million coins in 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, the Machine Learning model will predict the currency of the given coin.

### Types of Supervised Machine Learning

1. **Classification:** Classification is a supervised learning task where the output variable represents categories or classes. The output variable in classification is categorical, consisting of predefined classes such as "yes" or "no," "true" or "false," or other distinct categories. The goal of classification algorithms is to learn patterns from input features and assign the correct category to new instances. Common algorithms include decision trees, logistic regression, support vector machines, and neural networks, applied in diverse fields like spam detection and medical diagnosis.
2. **Regression:** Regression is a type of supervised learning where the goal is to predict a continuous or real-valued output variable. This output variable could represent quantities such as salary based on experience, weight based on height, or temperature based on time. Regression algorithms analyze the relationships between input variables and the continuous target variable to make predictions. Common regression techniques include linear regression, polynomial regression, and support vector regression, which are widely used in various fields such as finance, economics, and engineering for predictive modeling and forecasting tasks.

## Unsupervised Machine Learning

Unsupervised Machine Learning operates on unlabeled data, where there are no predetermined output variables. Instead, the model identifies patterns and structures within the data autonomously, without explicit guidance. By clustering similar data points or detecting underlying relationships, the model categorizes or organizes the data based on intrinsic features. This approach is particularly useful for tasks like data exploration, dimensionality reduction, and anomaly detection, where uncovering hidden structures is paramount. Here is how an unsupervised algorithm works:

1. You feed it an example input (without the associated output).
2. You repeat the above step many times. Eventually, the algorithm clusters your inputs into groups.
3. Now, you can feed it a new input, and the algorithm will predict which cluster it belongs to.

**Training Process:** In the training process of Unsupervised Learning, algorithms operate without external guidance or labeled data, learning autonomously to discern patterns and relationships within the input data. Unlike supervised learning, no explicit training is provided to the model, which instead relies on inherent structures within the data to make predictions or classifications. Primarily, unsupervised learning algorithms excel at tackling clustering and association problems, where the goal is to identify groups or patterns inherent in the dataset without predefined labels or categories.

**Applications of Unsupervised Learning**

1. Customer Segmentation.
2. Customer Churn Analysis.
3. Market Basket Analysis.
4. Semantic Clustering.

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

1. K Means Clustering.
2. Hierarchical Clustering.
3. DBSCAN.
4. 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 scoreless runs- bowler).

### Types of Unsupervised Machine Learning

1. **Clustering:** Clustering categorizes input data into groups based on patterns, like identifying customers with similar purchasing habits. The algorithm organizes data points into clusters without predefined labels, aiming to maximize intra-cluster similarity. This process aids in customer segmentation, anomaly detection, and pattern recognition by revealing natural groupings. Iterative refinement adjusts cluster assignments to optimize grouping accuracy and facilitate insights extraction.

## Reinforcement Machine Learning

Reinforcement Learning involves an agent interacting with an environment to learn through trial and error. The agent receives rewards based on its actions and aims to maximize its cumulative reward over time. Unlike supervised learning, there's no predefined target variable; the agent learns through exploration and exploitation. It's commonly used in tasks where the optimal action isn't explicitly known, such as game playing or robotics.

**Training Process:** Reinforcement learning operates on a reward-based system, where an agent interacts with an environment and receives feedback based on its actions. Through trial and error, the agent learns to maximize cumulative rewards by refining its decision-making strategies. Unlike supervised learning, there is no explicit target output; instead, the agent aims to optimize long-term performance. This approach is commonly employed in dynamic environments where actions influence future states and outcomes.

**Applications of Reinforcement Learning**

1. Building Games.
2. Training Robots.

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

1. Q-Learning.
2. Monte Carlo.

## Machine Learning vs Traditional Programming

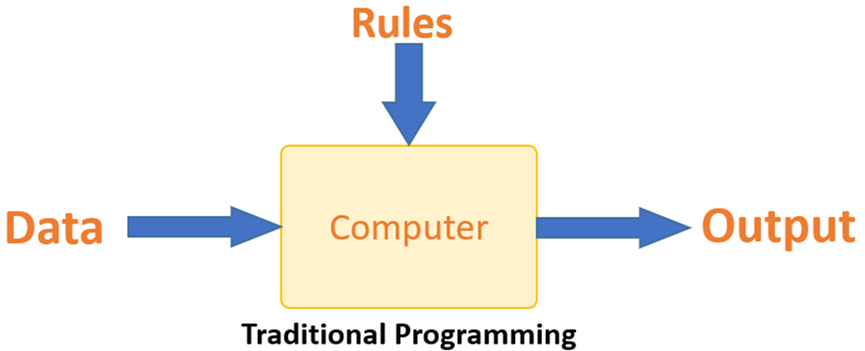
Traditional programming differs significantly from machine learning. In traditional programming, a programmer codes all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following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

Figure 1.1 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.

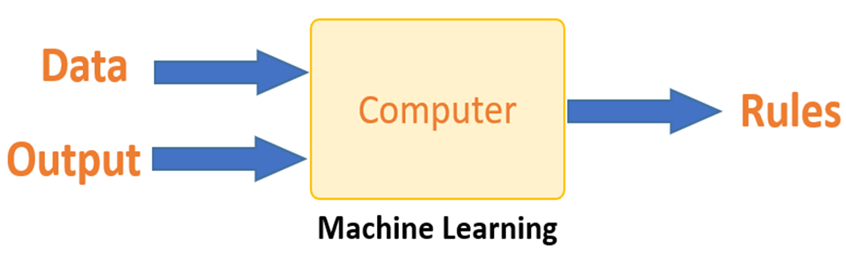


Figure 1.2 Machine Learning

## 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. 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 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 1.3 Learning Phase

For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood of going to a fancy restaurant. It turns out the machine finds a positive relationship between wages 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 the model again. You can use the model previously trained to make inference on new data.

The life cycle 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 d-g until the results are satisfying.
9. Use the model to make a prediction.



Figure 1.4 Inference from Model

### 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:

1. **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.

1. **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.

1. **Interpretability and Explainability**

**Challenge:** Many machine learning models, particularly deep learning models, are often considered "black boxes" due to their complexity, making it difficult to understand how they arrive at their predictions. This lack of interpretability poses a significant challenge in critical applications where understanding the reasoning behind model decisions is essential for trust and accountability.

**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).

1. **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.

1. **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.

1. **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.

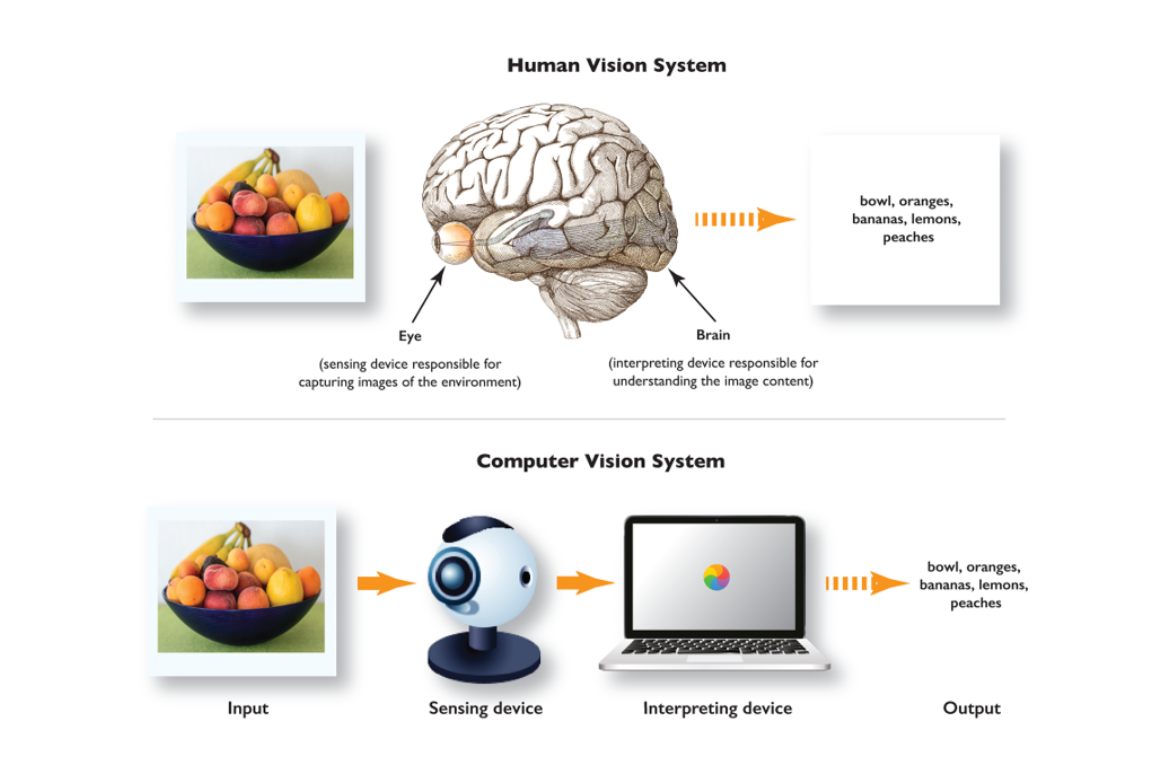


Figure 1.5 Human Vision vs Computer Vision

### How Computer Vision Works?

Generally, computer vision works according to the following basic steps:

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

Step 2: Processing the image.

Step 3: Understanding the image.

Step 4: Automation logic.

**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 algorithms 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.

Modern computer vision systems combine image processing with machine learning and deep learning techniques. Hence, developers combine different software (OpenCV or Open VINO) 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.

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

Figure 1.6 Computer Vision Application

1. **Facial recognition:** Utilizes computer vision to identify individuals by analyzing unique facial features, enabling applications like unlocking smartphones or enhancing security systems.
2. **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.
3. **Healthcare:** Utilizes computer vision for medical imaging analysis, aiding in accurate diagnoses and treatment planning for various medical conditions.
4. **Augmented Reality (AR):** Integrates computer vision to overlay digital information onto real-world environments, enhancing experiences like measuring objects using a smartphone camera.
5. **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.
6. **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 processing involves manipulating and analyzing digital images to extract useful information or enhance their visual quality. This field encompasses various techniques for representing images, filtering and enhancing their appearance, as well as detecting and extracting important features. Through these fundamental processes, image processing plays a vital role in applications ranging from medical imaging to computer vision and beyond.

1. **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.

1. **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.

1. **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 Techniques

Image classification, object detection, and segmentation are fundamental tasks in computer vision, each serving unique roles in visual data analysis. Image classification categorizes images into predefined classes, aiding in efficient organization and retrieval. Object detection identifies and locates specific objects within images, crucial for applications like autonomous navigation. Segmentation divides images into meaningful regions, enabling detailed analysis and scene understanding. These techniques are pivotal across domains, from healthcare to surveillance, driving advancements in AI and computer vision.

1. **Image Classification**

Image classification is the simplest technique of Computer vision (CV). The main aim of image classification is to classify the image into one or more different categories. The 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 color, item positions, etc., and for this, we need to go for any other CV technique.

Image classification encompasses two primary types: binary classification and multi-class classification. In binary classification, the goal is to identify whether a single class is present in the image or not, yielding results based on the presence or absence of the specified object. For instance, AI systems can achieve exceptional accuracy in detecting skin cancer by training on images containing both cancerous and non-cancerous skin samples. Multi-class classification, on the other hand, involves identifying multiple classes or categories within an image, enabling the system to distinguish between various objects or entities simultaneously.

1. **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, YOLOv2, etc. The choice of technique depends on factors like accuracy, speed, and computational resources. Each technique has its strengths and weaknesses, making it suitable for specific use cases and scenarios.

1. **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 colors also.

### Challenges and Future Directions in Computer Vision

Computer vision, an interdisciplinary field, encounters challenges and explores prospects. Despite advancements, issues like interpretability and robustness persist. Future directions aim to tackle these challenges and expand into new domains like healthcare and robotics. As technology evolves, computer vision systems become more adept at interpreting visual data. However, scalability and interpretability remain key concerns in this rapidly evolving field. Efforts in research continue to push the boundaries of computer vision, opening new avenues for innovation and application.

1. **Challenges in Computer Vision**

Challenges in computer vision persist despite notable advancements, requiring attention for continued progress. Key hurdles include interpretability of deep learning models, ensuring robustness across diverse datasets, and addressing ethical considerations like bias and privacy. Additionally, scalability remains a concern, particularly in real-time applications and large-scale deployments. Overcoming these challenges is imperative for unlocking the full potential of computer vision across various domains.

**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.

1. **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 the cities more liveable.

## Scope of The Project

We are 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 are 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"**

This paper discusses 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.

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

This paper discusses the into the intricacies of designing and implementing a smart car parking system leveraging IoT technologies. By integrating sensors and microcontrollers, the system continuously monitors parking space occupancy in real-time. Its primary aim is to furnish users with accurate information about available parking spaces, thereby significantly improving parking efficiency and minimizing search time. This study contributes significantly to advancing the realm of IoT-based parking solutions, addressing crucial challenges in urban mobility management.

1. **"Automated Car Parking System using Internet of Things”**

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

In system analysis, we examined the current system, outlining its methodology and identifying its limitations. This involved evaluating how data flows within the system, assessing its performance, and pinpointing areas for improvement. By analyzing the existing system's drawbacks, we could propose enhancements to optimize its functionality and efficiency.

## Existing System

The present system 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 present system provides 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

1. **Limited Accuracy:** IR sensors may have constraints in accurately detecting parked vehicles, leading to potential false positives or negatives.
2. **Dependency on Hardware:** The system heavily relies on physical components like IR sensors and microcontrollers, making it susceptible to malfunctions or damage.
3. **Limited Scalability:** Scaling the system to cover larger parking areas with numerous slots may pose challenges in terms of cost and complexity.
4. **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

The 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 Is Trained?

Training a machine learning model involves crucial steps to ensure its efficacy. Initially, data preparation involves collecting and preprocessing datasets to optimize them for training. Following this, the data is split into training and testing sets to assess model performance. Subsequently, the appropriate model architecture or algorithm is selected based on the task requirements. Finally, the model's performance is evaluated using various metrics to gauge its effectiveness and reliability. These systematic steps are integral to the successful training of machine learning models for diverse applications.

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 A black rectangular object with white text

Description automatically generatedmodel.

Figure 3.1 Empty Dataset

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

Figure 3.2 Not\_empty Dataset

1. **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.

1. **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.

1. **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

1. **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?

In the realm of machine learning, understanding how a model works is crucial for its effective deployment and utilization. This section delves into the key tasks and processes that underpin the functionality and efficiency of the model. By elucidating the intricacies of its operations, stakeholders gain valuable insights into the inner workings of the model, facilitating informed decision-making and optimization efforts.

1. **Image and Video Upload:** Users upload an image containing a mask defining the parking spots and a video showing the parking area.
2. **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.
3. **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.
4. **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.
5. **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.
6. **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

In the implementation phase, the software requirements and the necessary libraries are identified to develop and deploy the system effectively. These requirements include the programming language, frameworks, and tools needed to build the system. Additionally, specific libraries and packages are selected based on their compatibility with the chosen programming language and their functionality in fulfilling the system's objectives. By ensuring that all necessary software components are available and properly configured, the implementation process can proceed smoothly, leading to the successful development of the system.

## Software Requirements

In this project 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 :

1. Flask
2. OpenCV-Python
3. Scikit-learn
4. Numpy
5. Pillow
6. Scikit-image
7. 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 :

1. **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.
2. **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.
3. **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:

1. **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.
2. **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.
3. **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.
4. **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. Drawing on images also allows users to add different frame formats for clear visualization. This makes the visualization of the output look more clearer.

### Scikit-learn

Scikit-learn is a popular open-source machine learning library forclearert 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:

1. **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.
2. **Model Training:** The project employs scikit-lean’s Support Vector Machine (SVM) classifier (sklearn.svm.SVC) to build a model for classifying parking spaces.
3. 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 ( gamma and C values) using cross-validation.
4. **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.
5. **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:

1. **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.
2. **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.
3. 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.
4. **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.
5. **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 images 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

1. **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.
2. **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.
3. **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.
4. **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:

1. **Loading Images:** scikit-image can be used to load images from files into NumPy arrays, which can then be processed or analyzed further.
2. **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.
3. **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.
4. **Segmentation:** The library offers algorithms for image segmentation, which divides an image into regions or objects based on certain criteria. This could be useful for identifying parking spots or other regions of interest in the project.
5. **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.
6. **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:

1. **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.
2. **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.
3. **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.

## GitHub Repository Having Project Code

Refer the project code through the github link <https://github.com/LakshmiSahithii/CarParkingSpaceDetection1>. The Repository contains the main code file, model file and the zip file of the dataset.

# System Design

In system design, various diagrams aid in identifying and addressing potential issues or gaps in the system design early in the development process. By visualizing user interactions and system behaviours, stakeholders can better understand the system's functionality and validate its alignment with requirements. Moreover, use case diagrams provide a high-level view of the system's functionality, illustrating how users interact with the system to accomplish specific goals or tasks.

Similarly, class diagrams depict the static structure of the system, showcasing the relationships and attributes of classes and objects. Sequence diagrams offer a dynamic view of system behaviour, detailing the sequence of interactions between objects over time. State charts capture the various states and transitions of a system or component, while activity diagrams illustrate the flow of activities or processes within the system.

Together, these diagrams provide a holistic view of the system, guiding developers in implementing robust and scalable software solutions. Their visual representation enhances communication among project stakeholders, fostering collaboration and ensuring that the final product meets user needs and expectations. By leveraging these diagrams throughout the system design process, teams can streamline development efforts and deliver high-quality software solutions effectively.

## Usecase Diagram

A usecase diagram illustrates the interactions between actors and the system, depicting user-system scenarios. It showcases user actions and system functionalities from a user's perspective. Actors represent users or external entities, while use cases depict system features. These diagrams aid in understanding system behaviour and requirements. They are crucial in software development for capturing user needs and designing system functionalities. The Figure 5.1 shows the usecase representation of the system.

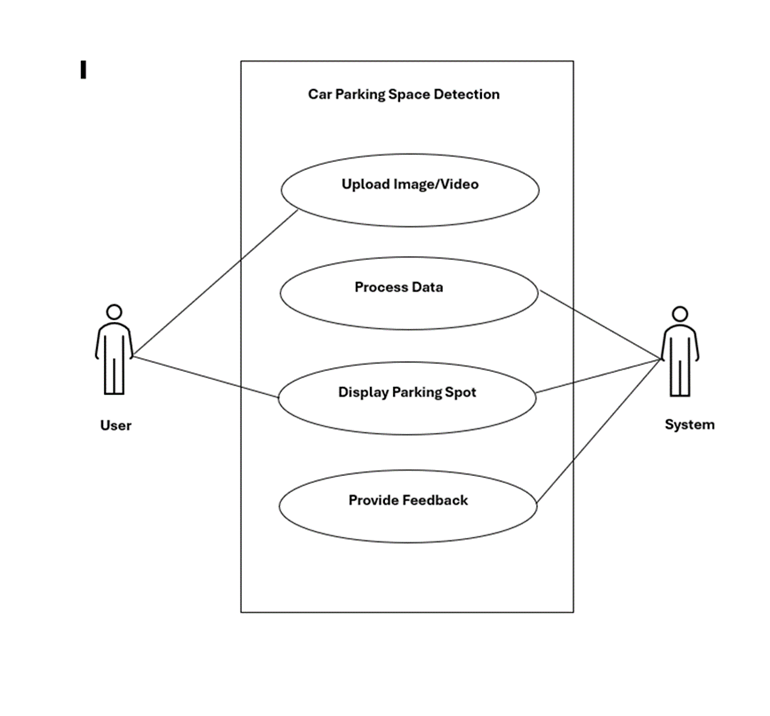


Figure 5.1 Usecase Diagram

## Class Diagram

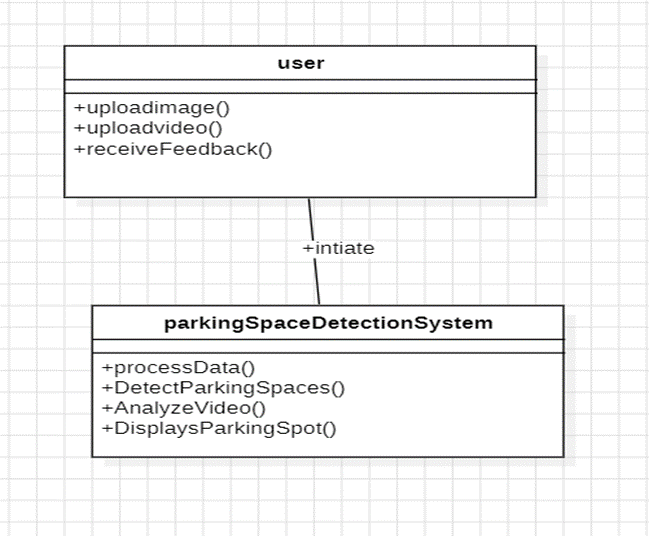
Class diagrams offer a visual depiction of the system's structure, detailing classes, attributes, and relationships. They act as a blueprint for software development, aiding developers in writing organized and scalable code. By illustrating the organization of classes and their associations, class diagrams enhance communication among project stakeholders. Additionally, they facilitate code generation and implementation, streamlining the development process. With their concise representation of entities and interactions, class diagrams promote collaboration and ensure the coherence and reliability of the system design. The Figure 5.2 shows the class diagram representation of the system.

Figure 5.2 Class Diagram

## Activity Diagram

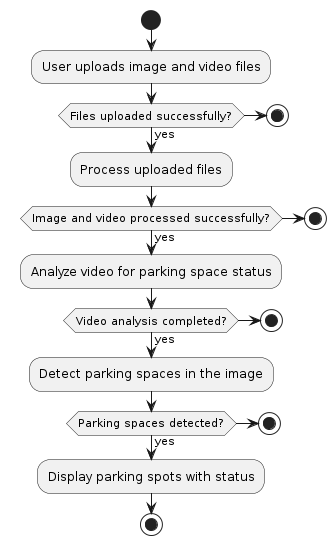
Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. In UML, an activity diagram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

Figure 5.3 Activity Diagram

## 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.

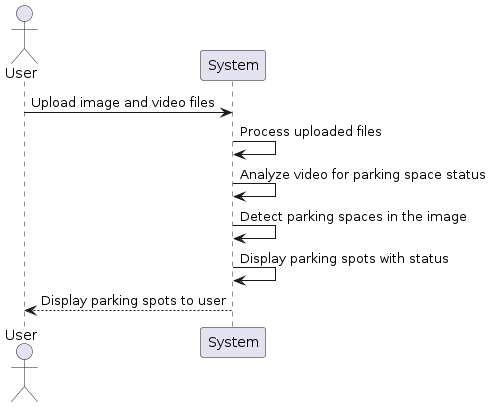


Figure 5.4 Sequence Diagram

## State Chart Diagram

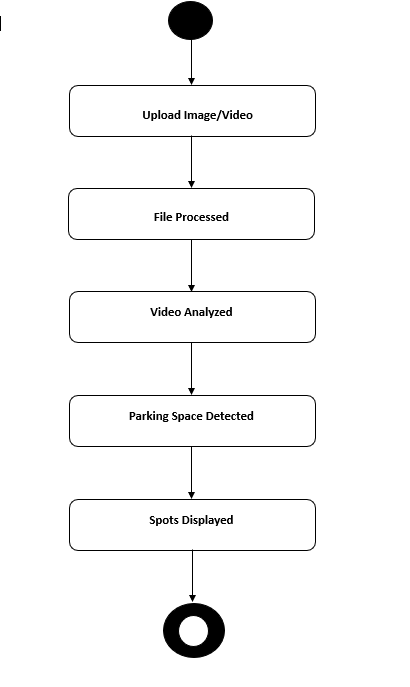
A state diagram, also known as a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modelling Language (UML). The Figure 5.5 shows the state chart diagram representation of the system.

Figure 5.5 State Chart Diagram

# Results

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

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

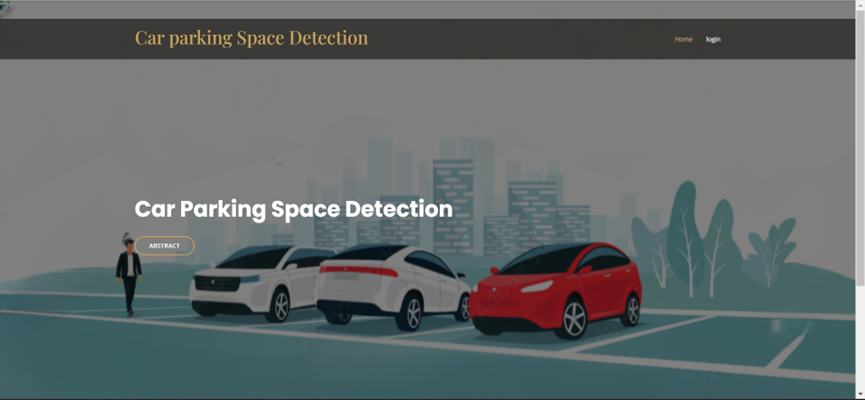
Home page of the website.

Figure 6.1 Home Page

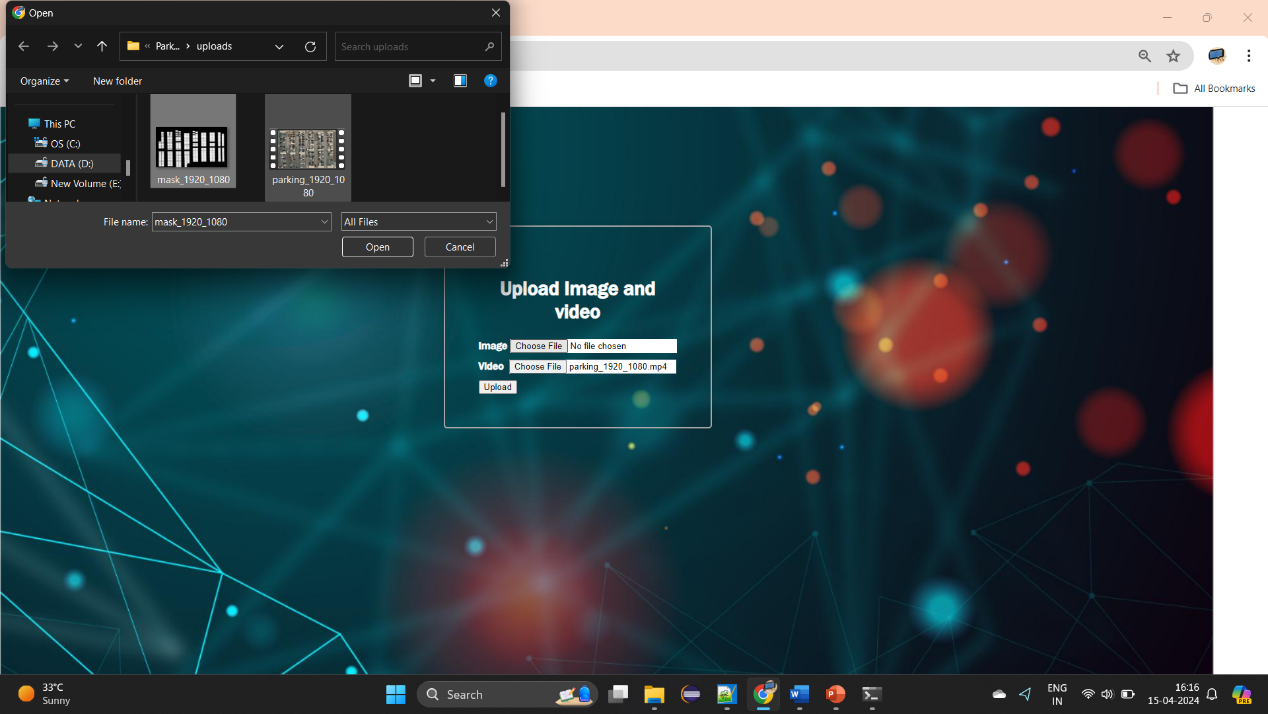
After Successful login into Website.

Figure 6.2 Upload Page

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

Figure 6.3 Result Page

# Conclusions

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.

# Bibliography

|  |  |
| --- | --- |
| [1] | P. Patil, S. Padaganur, Maruf Raza Gokak and K. Mirajkar, “IOT Based Car Parking”, IEEE, Bangalore, 2020. |
| [2] | A. Rashid, S. M. A. and I. M. S, “Design and Development of Smart Car Parking System using Internet of Things”, International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2),    pp. 23-27, 2018. |
| [3] | S. S. Shaikh and M. Deshpande, “Automated Car Parking System using Internet of Things”, International Conference on Sustainable Computing and Advanced Artificial Intelligence (ICSCAA), vol. 3, pp. 45-49, 2019. |
| [4] | P. Maheshwari and S. Tyagi, “Smart Car Parking System using Internet of Things”, International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), vol. 4, pp. 67-71, 2019. |
| [5] | J. Zhang, L. Ma, H. Jiang, S. Cao and H. Wang, “A Vehicle Detection Method Based on Deep Learning in Parking Lot”, IEEE International Conference on Image Processing (ICIP), vol. 5, pp. 123-127, 2019. |
| [6] | C. L. Hsu and C. L. Hsu, “A Fast and Low-Cost Detection Method of Empty Parking Spaces Based on Convolutional Neural Networks”, IEEE 5th International Conference on Computer and Communications (ICCC), vol. 6, pp. 89-93, 2019. |
| [7] | K. Bhuvaneswari and N. Krishnaraj, “Real-Time Car Parking Slot Detection System Using OpenCV”, International Conference on Advanced Computing and Communication Systems (ICACCS, vol. 7, pp. 145-149, 2020. |
| [8] | C. Guo, Y. Yao and J. Wang, “An Object Detection Method of Parking Lot Based on Single Shot MultiBox Detector,” IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA, vol. 8, pp. 201-205, 2020. |
| [9] | Y. Hu, Y. Zhu, J. Chen and Y. Shang, “Research on Car Parking Space Detection Algorithm Based on Deep Learning”, International Conference on Artificial Intelligence and Big Data (ICAIBD), vol. 9, pp. 67-71, 2021. |
| [10] | J. Chen and L. Wu, “Vehicle Detection and Classification for Intelligent Transportation System Based on Deep Learning”, 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), vol. 10, pp. 31-35, 2020. |
| [11] | H. Zhang, L. Liu and M. Zhang, “Vehicle Detection and Recognition Based on Deep Learning in Intelligent Traffic System”, 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), vol. 11, pp. 87-91, 2019. |