Parking Occupancy Detection Project Report

Prepared By,
Vinayak Kumar

Introduction:
The Parking Occupancy Detection Project leverages computer vision and machine learning techniques to automate the monitoring of parking spaces. The goal is to develop a system capable of determining the occupancy status of parking slots in outdoor environments. This report provides an overview of the project, outlining the key objectives, methodology, and outcomes.

Objectives:

1)Data Collection:
Acquire a relevant dataset containing images of parking spaces with labeled occupancy status.
Explore and understand the dataset's characteristics.

2)Data Preprocessing:
Load and preprocess the dataset, ensuring data integrity.
Resize and normalize images for consistency.
Explore a subset of the data for testing purposes.

3)Model Development:
Design and implement a Convolutional Neural Network (CNN) model to learn spatial features from parking space images.
Explore different model architectures and choose an appropriate one.

4)Model Training:
Split the dataset into training and testing sets.
Train the CNN model on the training set to learn patterns associated with occupied and vacant parking slots.

5)Application to Full-Scale Images:
Select a random full-scale image from the dataset.
Implement a mechanism to visualize parking slot status using bounding boxes.

Literature Survey:

In the endeavor to develop an advanced parking occupancy detection system, a thorough literature survey was conducted to investigate diverse methodologies and approaches employed by researchers in the field of computer vision and object detection. The objective was to gain insights into existing techniques, their efficacy, and potential enhancements for the proposed project.

1. Parking Space Detection in Different Weather Conditions Based on YOLOv5 Method (ICSECS 023 IEEE 8th International Conference On Software Engineering and Computer Systems):
The paper titled "Parking Space Detection in Different Weather Conditions Based on YOLOv5 Method" presents a novel approach to parking space detection, focusing particularly on addressing challenges posed by varying weather conditions. The authors leverage the YOLOv5 model, a cutting-edge object detection methodology known for its real-time processing capabilities and high accuracy. It is noteworthy that the study utilizes the CNRPark+EXT dataset, a dataset we also employ in our project.

The research likely delves into the impact of weather conditions, such as rain or sunshine, on the performance of parking space detection algorithms. The choice of YOLOv5 suggests an emphasis on efficiency and effectiveness in real-world scenarios, aligning with our project's objectives.

Just Like YOLOv5 uses boxes, I Have created my own red and green boxes to represent a place is parked or vacant respectively.

Methodology:

1)Data Loading and Preprocessing:
The project starts by loading the CNRPark+EXT dataset, a collection of images with corresponding occupancy labels. A smaller subset of the data is extracted for testing purposes. The images are loaded using OpenCV, resized for consistency, and checked for valid dimensions.

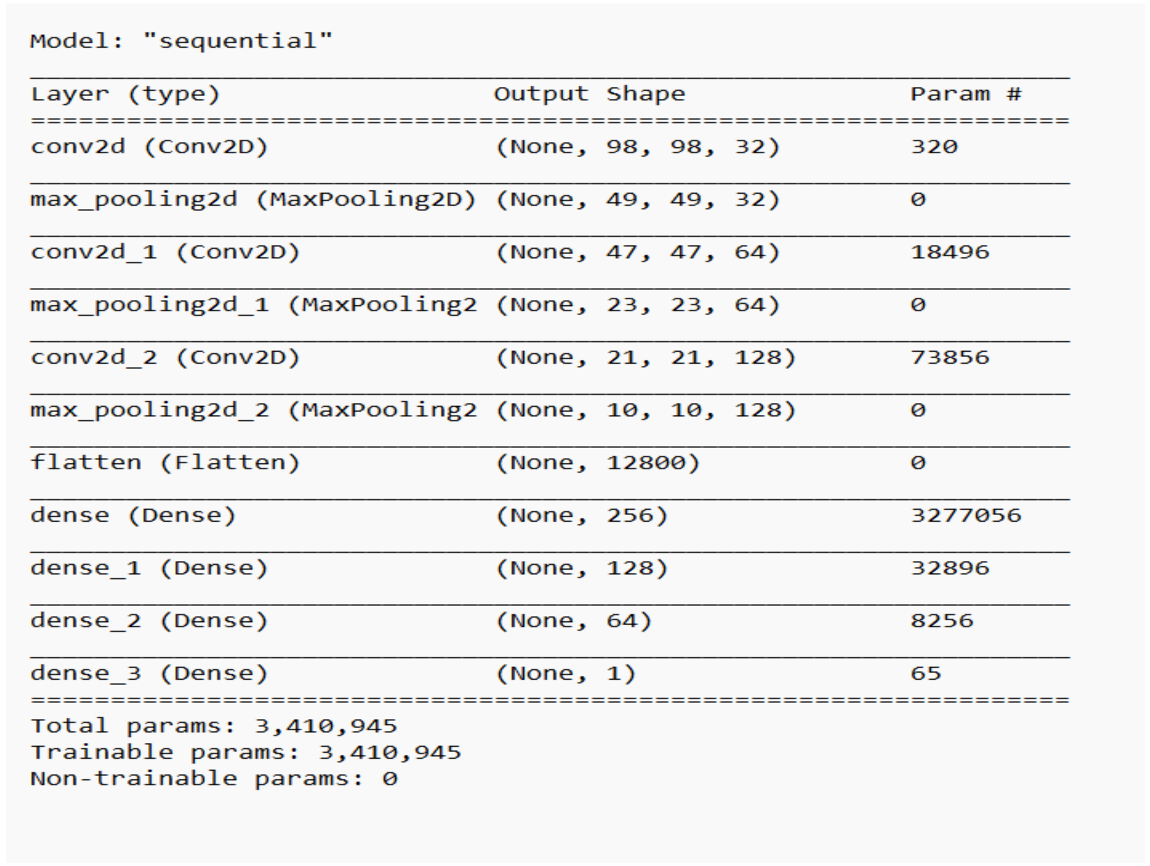
2)Model Development:
A Convolutional Neural Network (CNN) model is constructed using TensorFlow's Keras API. The chosen architecture consists of multiple convolutional layers followed by max-pooling layers to capture spatial features. Dense layers are employed to capture global patterns, and the final layer provides a binary output indicating parking slot occupancy.

3)Model Training and Evaluation:
The dataset is split into training and testing sets, and pixel values of images are normalized. The CNN model is trained on the training set for a specified number of epochs. The trained model is then evaluated on the test set to assess its accuracy and performance.

4)Application to Full-Scale Images:
A random full-scale image is selected from the dataset. Bounding boxes are drawn on the image based on patch information from a corresponding CSV file. The CNN model is applied to predict parking slot occupancy within each patch, and the result is visualized by coloring the bounding boxes.

CNN Model Architecture:

The CNN model comprises convolutional layers, max-pooling layers, and dense layers. The input layer processes 100x100 grayscale images, and the model gradually learns hierarchical features. The larger model chosen for this project has a total of 3,410,945 parameters, allowing it to capture more complex patterns.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 98, 98, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 49, 49, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 47, 47, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 21, 21, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 12800)             0
_________________________________________________________________
dense (Dense)                (None, 256)               3277056
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65
=================================================================
Total params: 3,410,945
Trainable params: 3,410,945
Non-trainable params: 0
```

Code Explanation:

1)Importing Libraries:
The code starts by importing necessary libraries such as OpenCV, NumPy, Pandas, scikit-learn, and TensorFlow.

2)Loading Dataset:
The dataset is loaded from a CSV file named "CNRPark+EXT.csv" into a Pandas DataFrame (data).

3)Data Preprocessing:
A smaller subset (data_subset) of 500 random samples is extracted for testing purposes.
Features (images) and labels are extracted from the subset by iterating through the rows of the DataFrame.

4)Data Splitting and Normalization:
The dataset is split into training and testing sets using scikit-learn's train_test_split.
Pixel values of images are normalized to the range [0, 1].

5)Building a CNN Model:
A larger Convolutional Neural Network (CNN) model is defined using TensorFlow's Keras API.
The model consists of convolutional layers, max-pooling layers, and dense (fully connected) layers.
Binary cross-entropy is used as the loss function, and Adam optimizer is used for training.

6)Training the Model:
The larger CNN model is trained on the training set for 10 epochs.

7)Evaluating Model Performance:
The model's accuracy is evaluated on the test set.

8)Random Full-Scale Image Selection:
A random index is chosen to select a full-scale image from the dataset.

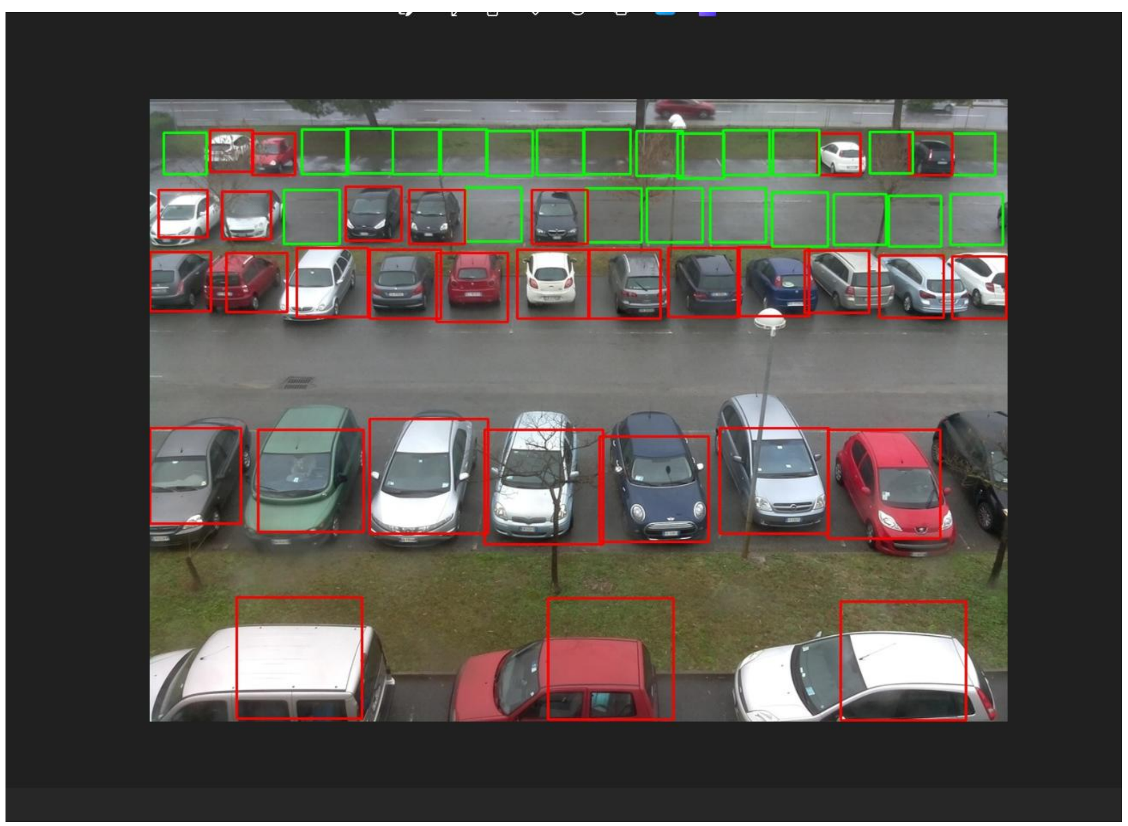
9)Creating Full Image Path:
The path to the selected full-scale image is constructed based on information from the dataset.

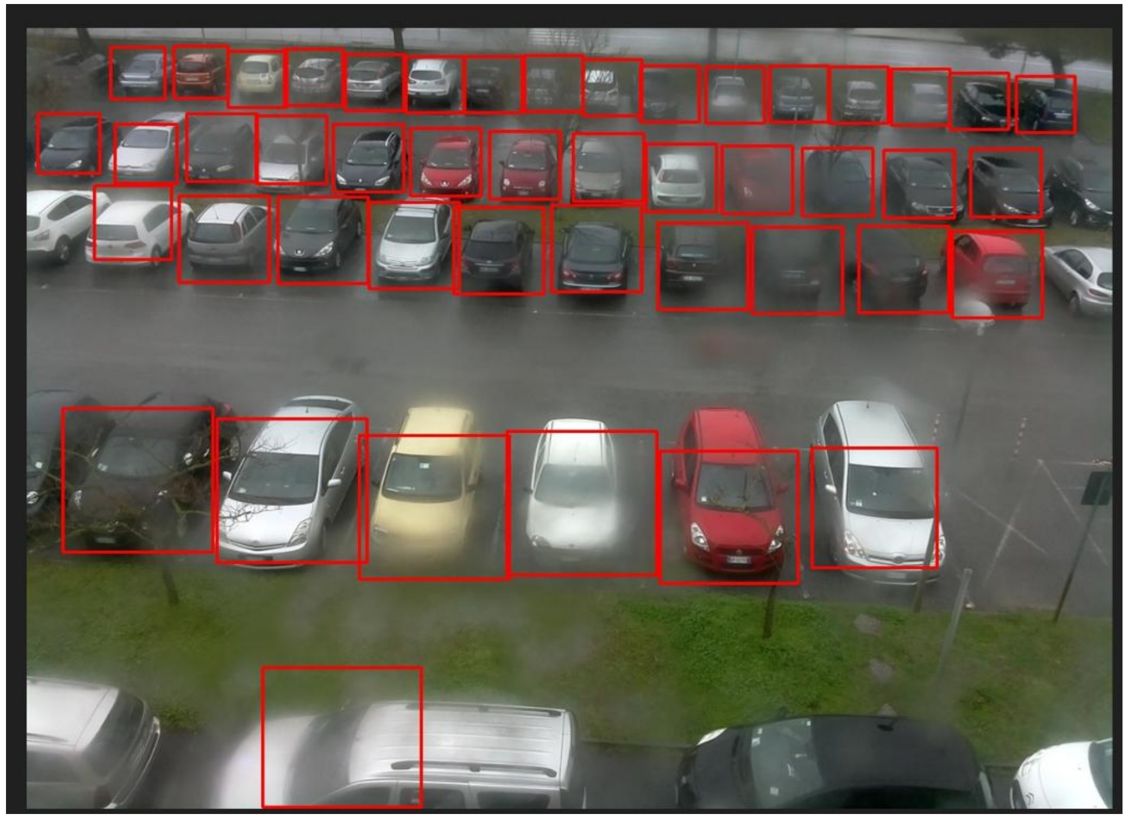
10)Visualizing Patches on Full Image:
The script loads the corresponding CSV file for the selected camera and reads patch information.
It draws red boxes for occupied parking slots and green boxes for vacant slots on the full-scale image.

11)Saving and Displaying Result:
The resulting image with boxes indicating parking slot status is saved and displayed.

Results:

The trained model achieves a certain accuracy on the test set, indicating its effectiveness in predicting parking slot occupancy. The visualization on full-scale images provides a practical demonstration of the model's application to real-world scenarios.





Conclusion:

The Parking Occupancy Detection Project successfully demonstrates the utilization of computer vision and machine learning techniques for automating the monitoring of parking spaces. The developed CNN model proves effective in predicting parking slot occupancy, and the application to full-scale images showcases the potential for real-world implementation.