Ivan Franko National University of Lviv

University of L'Aquila

Double-Degree Master's Programme - InterMaths
Applied and Interdisciplinary Mathematics

Master of Science **Applied Mathematics**	**Master of Science** **Mathematical Engineering**
Ivan Franko National University of Lviv (IFNUL)	University of L'Aquila (UAQ)

Master's Thesis

DETECTING PARKING OCCUPANCY USING MACHINE LEARNING

Supervisor
Assoc. Prof. Yuriy Muzychuk

Candidate
Joshua Junior John

Co-advisor
Prof. Maciej Sablik

Student ID (UAQ): 274409
Student ID (IFNUL): 27210487C

Academic Year 2021/2022

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
IVAN FRANKO NATIONAL UNIVERSITY OF LVIV

Faculty of Applied Mathematics and Informatics
Department of Applied Mathematics

Master's Thesis

DETECTING PARKING OCCUPANCY USING MACHINE LEARNING

Author: student of group PMP-62
specialty 113 – Applied Mathematics

_____ John, Joshua J.

Supervisor:

_____ Muzychuk, Yuriy A.

Co-advisor:

_____ Sablik, Maciej

Lviv – 2022

TABLE OF CONTENTS

ABSTRACT

By giving real-time indicators of parking availability, parking occupancy detection has the potential to minimize traffic congestion in congested regions. Currently, such systems are largely used in interior contexts, and they rely on expensive sensor-based methodologies. Consequently, with a higher demand for parking space outdoor, low-cost image detection methods have become a focus of research and development recently. Driven by the notable performance of Convolutional Neural Networks (CNNs) in various image classification and recognition tasks, this study presents a robust parking occupancy detection framework by using a modified Mask RCNN with Intersection over Union (IoU) and a custom CNN Model based on transfer learning to detect the occupancy of outdoor parking spaces from images. The model was trained and tested with images from parking lots under different illuminance and weather conditions.

ABBREVATIONS

CNN - Convolutional Neural Networks

RCNN – Region Based Convolutional Neural Networks

MRCNN – Mask Region Based Convolutional Neural Networks

IoU – Intersection over Union

HSV – Hue Saturation Value

RGB – Red, Green and Blue

CCTV - Closed Circuit Television

YOLO – You Only Look Once

OpenCV – Open Computer Vision

COCO – Common Objects in Context

FCN – Fully Convolutional Network

API – Application Programming Interface

CHAPTER ONE

1. INTRODUCTION

In recent years, with rapid economic development and increase in standard of living, the ownership of automobiles has also increased year by year(Wu, Zhao and Ou, 2014). This increasing growth has also brought about stationary traffic and other problems. Stationary traffic refers to the parking of vehicles, including short-term parking due to passengers getting on and off or loading and unloading of goods, and long-term parking in parking lots. The problems of stationary traffic are mainly manifested in the following aspects: insufficient control of land use indicators, planning and layout, fewer parking spaces in public buildings, a severe road occupation occurrence, and fewer parking spaces in parking lots. The above situation demonstrates that the parking problem is very acute. As a result, detecting parking occupancy has risen to the top of the priority list for resolving parking issues (Wu, Zhao and Ou, 2014;).

1.1. Problem statement

The number of motor vehicles in the world has increased rapidly due to the fast economic and social development (Wu, Zhao and Ou, 2014). Building parking lots, on the other hand, has progressed at a glacial pace, and the issue of parking challenges has grown increasingly prevalent. Finding an available parking space nowadays is an issue that should not to be neglected, because amongst other factors, it consumes time and energy.

1.2. Aim of Research

This research paper focuses on mitigating the problems associated with finding an available parking space by applying machine learning technique.

1.3. Research Materials

The algorithms for implementation and visualization were written using Python Programming Language.

The following libraries were used:

1. Tensorflow - An open-source software library used for Machine Learning (Evermann, Rehse and Fettke, 2017)
2. Keras - An open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.(Chollet, 2015)
3. Numpy – A library for scientific computing (Oliphant, 2006).
4. Matplotlib – A library for plots and charts (Hunter, 2007)

1.4. Dataset Description

The dataset consists of 120,000 segmented images representing both the occupied and empty slot of a parking lot. The dataset was extracted from images in the PKLot dataset acquired from a parking lot in the Federal University of Parana (UFPR) located in Curitiba, Brazil (Almeida *et al.*, 2015). The image acquisition was made by a 5-minute time-lapse interval over 30 days during the daytime on three weather conditions namely sunny, rainy, and cloudy days as shown in the figures below.

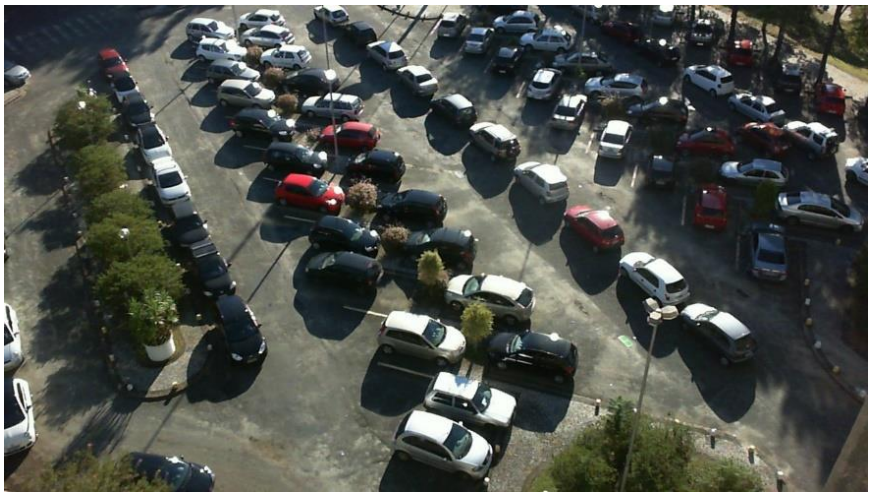

Figure 1: Parking Lot (Sunny Weather)

Figure 2: Parking Lot (Cloudy Weather)

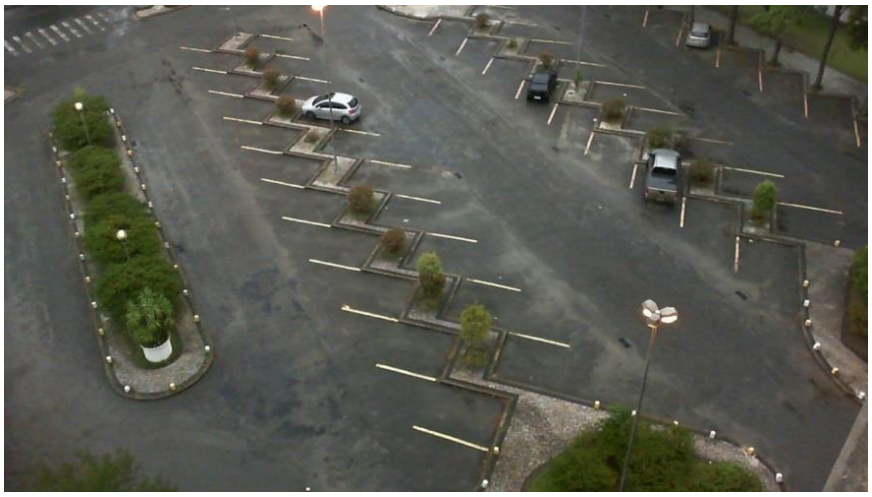

Figure 3: Parking Lot (Rainy Weather)

In addition to the above, a short clip of a mini outdoor parking lot was recorded using a smartphone's camera in the Lviv Oblast and also, an aerial footage of a parking lot in Dubai was used (Neiezhmakov, 2019).

Figure 4 Mini Parking Lot

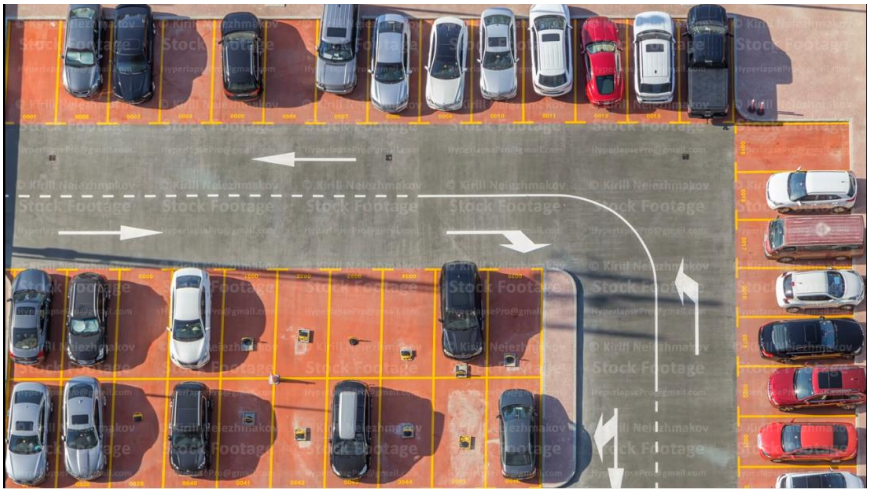

Figure 5 Aerial footage of parking lot (Neiezhmakov, 2019)

1.5. Practical Value of Results

The area and the city stand to profit from the detection of parking occupancy in a number of ways, including an improvement in traffic flow, a reduction in congestion, greater mobility, and enhanced living circumstances. It helps save time and energy, reduces traffic difficulties, and brings the levels of air and noise pollution down, further contributing to a major reduction in the levels of both types of pollution.

Keeping track of parking spaces' occupancies also has a number of positive social effects. Some of the direct and indirect social benefits include an increase in personal safety at night, a reduction in accidents and car damage, a minimization of theft, an increase in safety for pedestrians, the provision of accessible parking for all users, including the disabled and parents with children, a reduction in anxiety among drivers, which in turn reduces stress levels and eliminates the need for unnecessary vehicle speeding and honking, as well as other benefits. The alleviation of parking issues will be beneficial to local businesses, since it will lead to an increase in the number of customers walking through their doors. The residents will benefit from fewer cars driving in circles around them, which will result in time and money saved, less traffic congestion, and an overall rise in the quality of life (Vuk and Andročec, 2022)

1.6. Related and existing research

A lot of work has been done related to this field, for example: To identify a moving object entering the parking lot, a sensor-based technique can be utilized. The car's occupation of a place is inferred from the size of the object and the time it takes to cross the gate where the sensor is positioned (Lee, Yoon and Ghosh, 2008) . Sensors' application is extensive yet limited when money is a concern, due to the expense of installation and maintenance. Furthermore, they cannot be used outside because there is no roof on which to mount a sensor.

Another research in this field is the use of binary classifiers for Convolutional Neural Network (CNN) used for the smart parking system (Thomas and Bhatt, 2018). The data acquired is an RGB image converted to HSV to get the value of V, which is value. A transformation was done to the gray value and Second Derivative performed so that it becomes even broader with sharpened outlines. This process's results are converted into binary images and entered into the CNN. The research does not explain how the system can detect each empty or filled parking spaces.

Kumar *et al.*, proposed method to detect parking spaces availability by using a drone camera (Kumar *et al.*, 2021). First, the parking area detected by giving four points in each

parking area. After that, we need to make corrections on the result images to transform the lines created into a straight line to save its coordinates. To determine the availability of parking spaces, a line model used where the already parked car will form a straight line. If there is a parking space available, then the line will be disconnected, then the disconnected line will be marked as an available parking space. This research has an effective method for drones only, so if this method is applied to a camera, it will still have deficiencies in which it cannot form a line to show parking spaces availability.

An approach based on You Only Look Once (YOLO) V3 and Lite Alexnet has already been presented to speed up the identification of parking spot availability. The researchers used the YOLO V3 border box to detect parking spots automatically. Classification of parking space availability using CNN with Alexnet architecture has modified the number of layers and parameters called Lite Alexnet. This research has a constraint that is needed to help marks in the form of lines on the parking spaces to find the parking spaces from the whole image automatically (Tanuwijaya and Fatichah, 2020).

The authors presented a system for detecting parking spaces that is based on an ultrasonic sensor as part of the grid project. They installed ultrasonic sensors at intervals of around 5 meters in the ceiling in order to execute a technique for knowing the spatial position. However, the report only mentions a single obstruction to look out for, and there are no specifics provided on the range of vehicles that operate in that region. Because the ultrasonic employs the peak of the frequency beam into many angles into barriers, the precise placement of each parking spot should necessitate the use of other sensors in addition to this sensor. (Shao, Chen and Cao, 2018).

For this project, we intend to make improvements upon the above listed references. For our first approach, we would apply modified M-RCNN to auto detect parking lot regions and detect occupancy while our second approach would classify occupied and empty slots.

CHAPTER TWO

2. METHODOLOGY

For this research, we propose two approaches. First is the use of Mask RCNN to automatically detect the parking slots and also available parking slots. While the next is the use of a custom CNN model which is trained with dataset containing empty and occupied slots.

2.1. Mask Regional Convolutional Neural Network

Mask R-CNN extends Faster R-CNN to handle instance segmentation problems (He *et al.*, 2017). This is accomplished by adding an object mask prediction branch to the current branch for bounding box recognition. Mask R-CNN is an intuitive extension of Faster R-CNN in theory, but properly designing the mask branch is important for excellent results. Most notably, Faster R-CNN was not designed to match network inputs and outputs pixel-by-pixel. RoIPool, the de facto basic process for attending to instances, demonstrates this by doing coarse spatial quantization for feature extraction. Mask R-CNN corrects the misalignment by using RoIAlign, a basic, quantization-free layer that reliably maintains exact spatial positions (Naufal, Fatichah and Suciati, 2020).

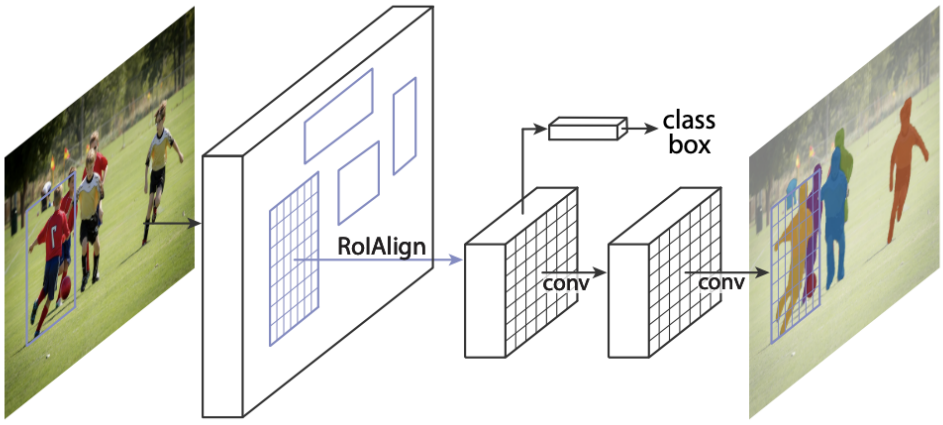

Figure 6: Mask RCNN (He *et al.*, 2017)

In order to detect parking occupancy, we simplify this approach into three stages which includes identification of region of interest, detection of cars and computing free parking areas using the Intersection over Union (IoU) algorithm as seen in Figure 7

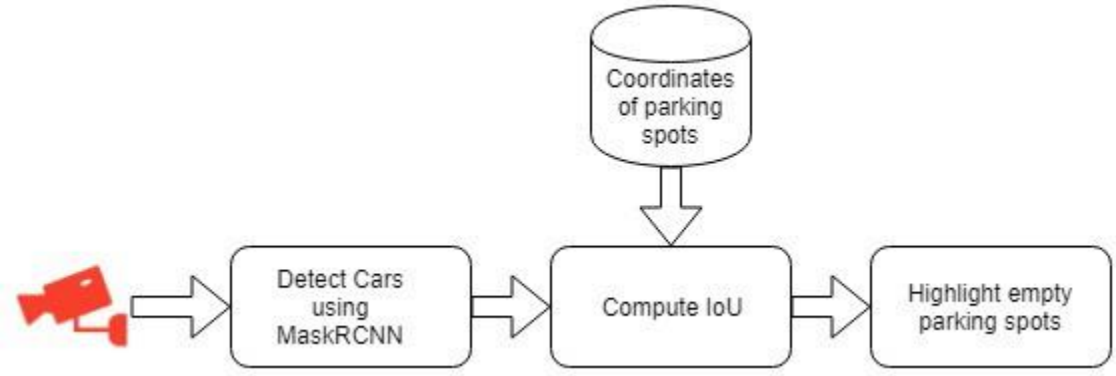

Figure 7: Flow process using M-RCNN approach

2.1.1. Identification of Region of Interest

The initial stage in a parking space identification system is to locate the region of interest, which is the total number of parking spaces. We take two approaches here; First, we get a frame of parking area of interest filled to its capacity, then we detect all vehicle in a parking region using use M-RCNN object detection model, this acquired coordinates serves as our region of interest. The first approach sometimes leads to false positives and true negatives. We take the second approach which involves marking out parking lot boundaries once by using a python script for marking out the slot as seen in Figure 8. This acquired coordinates serves as our region of interest

Figure 8 Marking of Parking Slots

2.1.2. Detecting Vehicles in a Video

The Mask-RCNN will be used to detect vehicles in the video. The bounding box coordinates, masks of identified items, confidence score for each prediction, and class ids of detected objects will all be returned by M-RCNN on every frame of the video. We'll now filter out the bounding boxes of the vehicles, trucks, and buses using the class ids.

2.1.3. Calculating Intersection Over Union (IoU)

It is the ratio of the overlap and intersecting areas, as the name implies. As a result, computing Intersection over Union may be done using the equation (2.1)

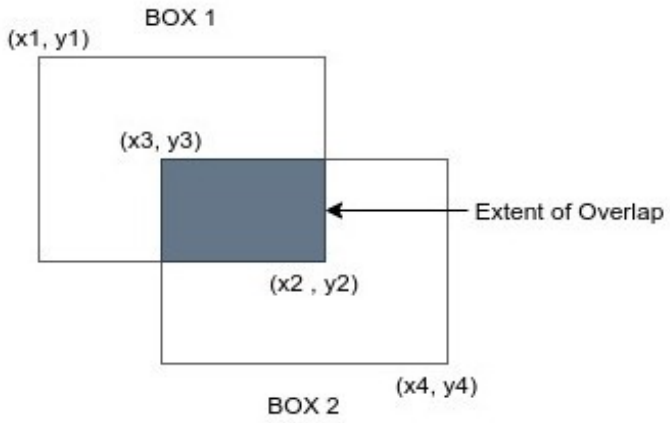

Figure 9: Intersection of two boxes

$$IoU = \frac{Area\ of\ Intersection\ of\ two\ boxes}{Area\ of\ Union\ of\ two\ boxes} \quad \textbf{(2.1)}$$

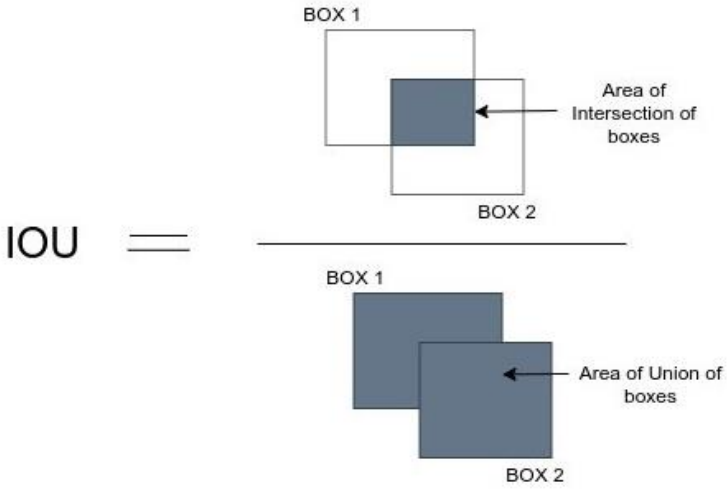

Figure 10 Diagrammatic Representation of the formula to calculate IOU

For each pair of parking slot coordinates and bounding box of vehicles, we calculate the IoU. If the IoU for a pair exceeds a particular threshold, that parking place is considered occupied; otherwise, it is empty.

2.2. Custom Convolutional Neural Network

Convolutional Neural Networks (CNN) is a deep learning approach effective for vision tasks. A CNN is made up of a potentially huge number of hidden layers, each of which conducts mathematical computations on the previous layer's input and creates an output that is fed into the next layer (Amato *et al.*, 2017b).

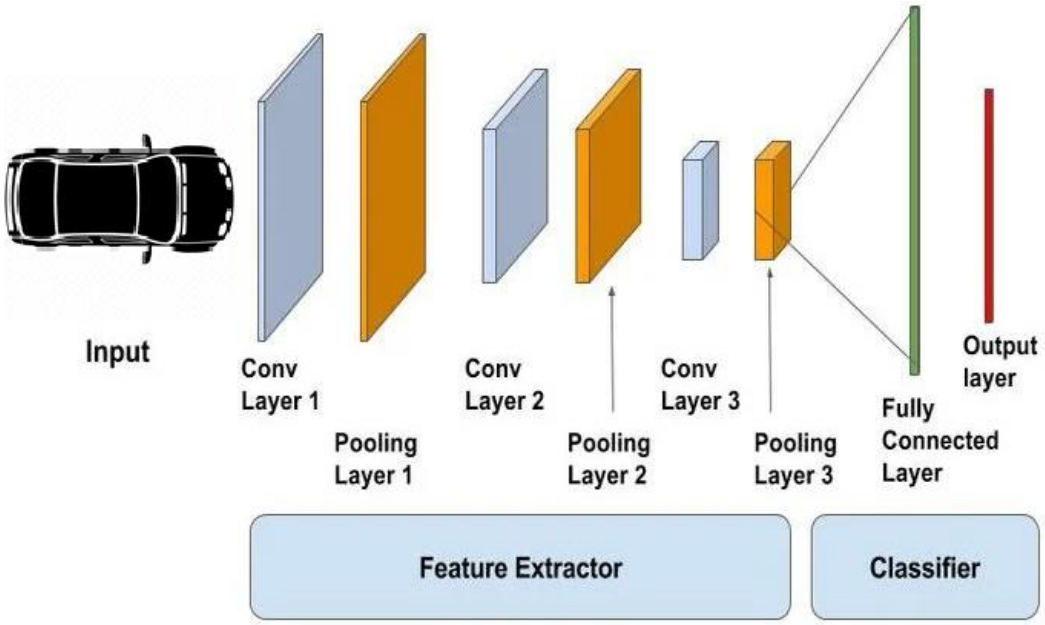

Figure 11 CNN Architecture

The existence of convolutional layers, which can better represent and identify the spatial correlation of surrounding pixels than conventional fully connected layers, distinguishes CNNs from classical neural networks. The final outputs of the CNN for a classification task are the classes that the network has been trained on. The training step is often quite computationally intensive and might take a long time to complete. Once the network has been trained and the classifier has been properly setup, the prediction run time is fairly fast and efficient.

For this approach we created a custom CNN model using stacks of Conv2D and MaxPooling2D layers. As input, a CNN takes tensors of shape (img_height, img_width, color_channel). Color channels refers to (R, G, B). Our CNN receives as inputs images of

shape (img_width, img_height, 3). We see that the output of every Conv2D and MaxPooling2D layer is a 3D tensor of shape (height, width, channels). The width and height dimensions tend to shrink as you go deeper into the network. The number of output channels for each Conv2D layer is controlled by the first argument (for example, 32 or 64). To complete the model, we feed the last output tensor from the convolutional base into one or more dense layers to perform classification. Dense layers take vectors as input (which are 1D). We flatten (or unroll) the 3D output to 1D, then add two dense layers with a sigmoid activation function connected to the output layer.

2.3. Transfer Learning

Transfer learning is a machine learning technique in which a model created for one task is utilized as the basis for a model for a different task. It is an optimization that allows rapid progress or improved performance when modelling the second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

A pre-trained model which can be used for transfer learning is the VGG16 Net. VGG stands for Visual Geometry Group (a group of researchers at Oxford). VGG- Network is a convolutional neural network model proposed by (Simonyan and Zisserman, 2015). This architecture as shown in Figure 12 achieved top-5 test accuracy of 92.7% in ImageNet, which has over 14 million images belonging to 1000 classes.

We applied transfer learning by creating a CNN model leveraging on the VGG16 Net to perform binary classification.

Figure 12 VGG16 Architecture (Simonyan and Zisserman, 2015)

CHAPTER THREE

3. RESULT

This chapter discusses the implementation and analyses the result obtained using both approaches to detecting parking using machine learning.

3.1. Mask-RCNN Approach:

a. Identification of Region of Interest: the region of the detected boxes serves as our region of interest

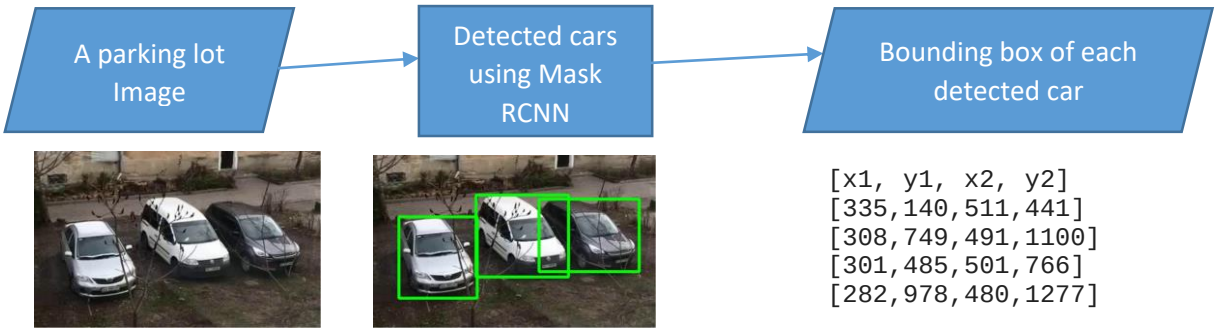

Figure 13: Detecting Vehicles Using Mask RCNN for a full parking spot

b. Detecting Vehicles when parking slot in not full capacity

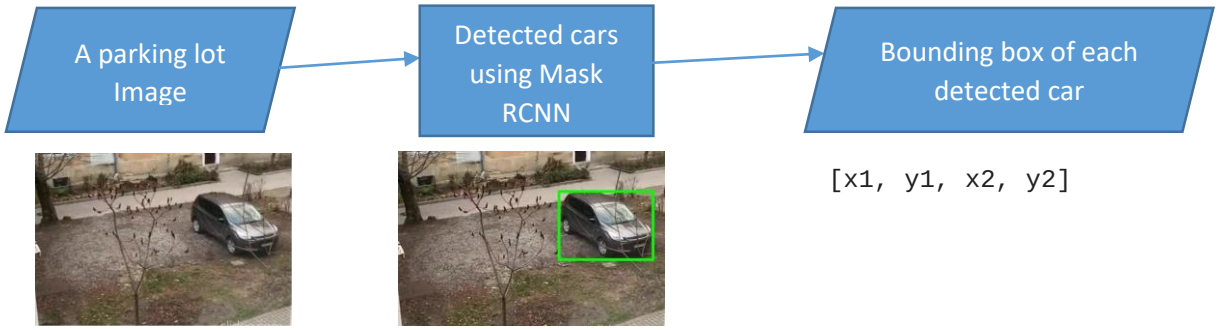

Figure 14: Detecting Vehicles Using Mask RCNN when the parking lot is not full

c. Applying IoU threshold to get free parking slot

Here we applied an IoU threshold of 0.15 to detect free parking space as seen below

Figure 15: Detected parking space

3.1.1. Performance metrics evaluation

In this research, we measure the performance of detection results quantitatively. we used various IoU value to calculate parking space detection accuracy, sensitivity and specificity. The accuracy, sensitivity, and specificity are defined in Equations (3.1), (3.2), and (3.3) respectively.

In the equations (3.1), (3.2), and (3.3), TP (True Positive) is the number of occupied sub-images classified as occupied, TN (True Negative) is the number of unoccupied sub-images classified as unoccupied, FP (False Positive) is the number of unoccupied sub-images classified as occupied, and FN (False Negative) is the number of occupied sub-image classified as unoccupied

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \textbf{(3.1)}$$

$$Sensivity\ (True\ Positive\ Rate) = \frac{TP}{TP+TN} \quad \textbf{(3.2)}$$

$$Specificity\ (True\ Negative\ Rate) = \frac{TN}{TP+TN} \quad \textbf{(3.3)}$$

3.1.2. Evaluation Results

The test was done on a sample of distinct frames/images extracted from a video capturing a mini-parking space.

The table and figure shows the plot of the accuracy at various IoU threshold values.

IoU Threshold	Accuracy	Sensitivity	Specificity
0.10	0.70	1.0	0.0
0.15	0.70	1.0	0.0
0.20	0.70	1.0	0.0
0.25	0.70	1.0	0.0
0.30	0.60	0.67	0.33
0.35	0.70	0.57	0.43
0.40	0.70	0.57	0.43
0.45	0.70	0.57	0.43

Table 1: IoU Threshold, Accuracy, Sensitivity and Specificity

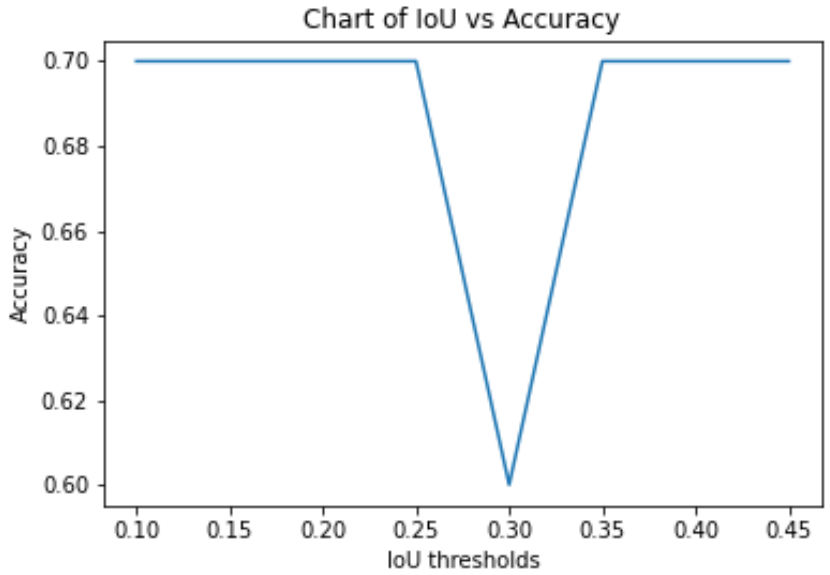

Figure 16: IoU vs Accuracy

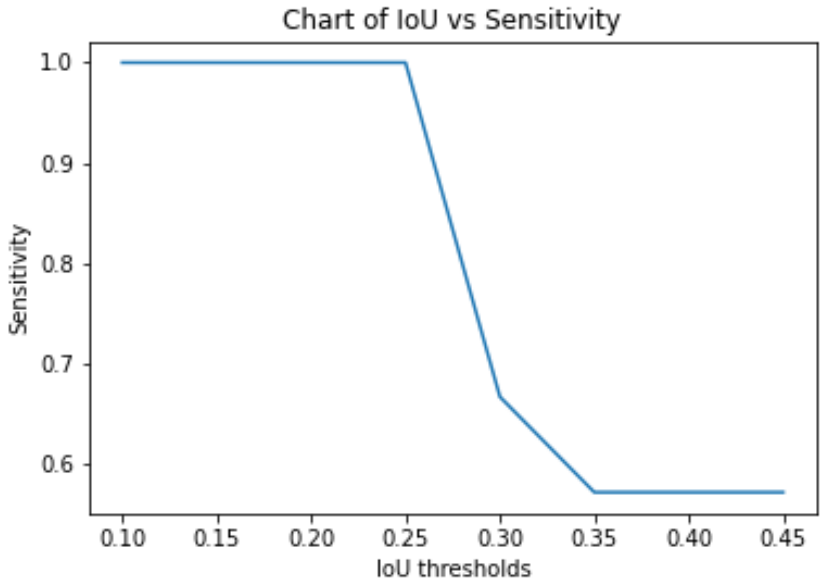

Figure 17: IoU vs Sensitivity

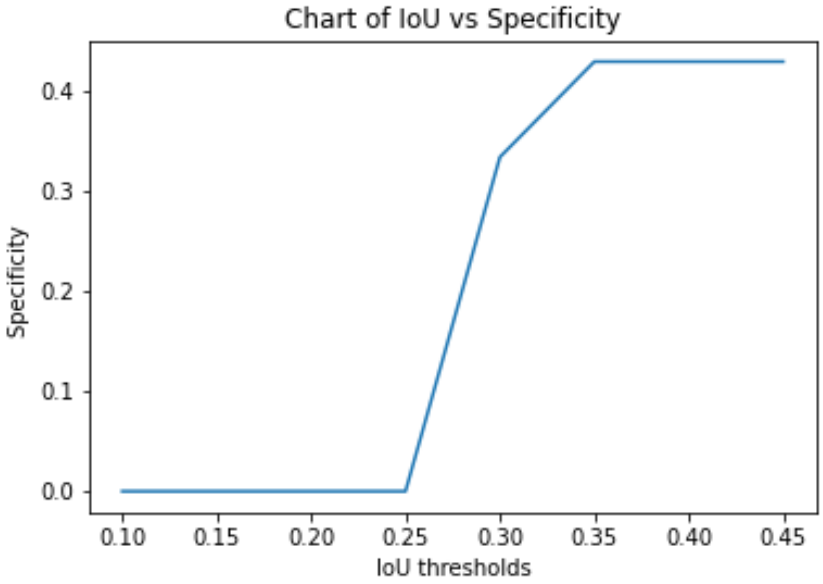

Figure 18: IoU vs Specificity

From this, we can see that the detector performed best at IoU threshold values less than 0.25. We can also see how sensitive the IoU value affects our result.

3.2. Custom CNN Model Approach

a. Data Description

As stated previously, our dataset contains 120,000 segmented images representing rainy cloudy and sunny weather. Also, our dataset includes the positions of the slots in the parking lot (Almeida *et al.*, 2015; Amato *et al.*, 2017a).

b. Data Visualization

We used 60,000 images for each class (empty and occupied). These images were split into training and test using a 70%, 30% respectively.

```
1 train_images, test_images, train_labels, test_labels = train_test_split(images, image_labels, test_size = 0.3, random_state = 42)
2 print(f'count of training images: {len(train_images)}')
3 print(f'count of test images: {len(test_images)}')
```

```
count of training images: 84000
count of test images: 36000
```

Figure 19 Training and Test Images Count

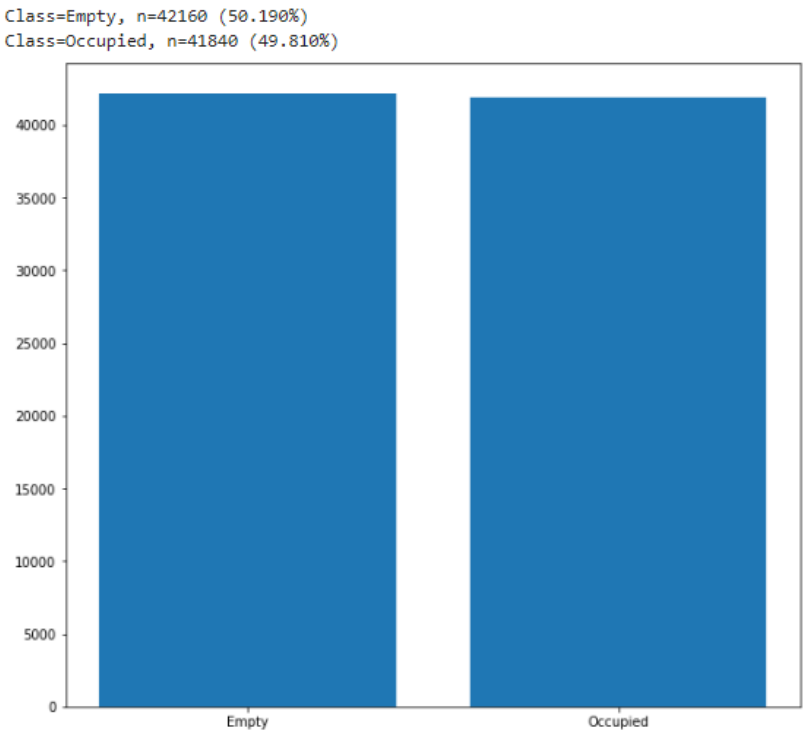

Figure 20 Empty vs Occupied data distribution

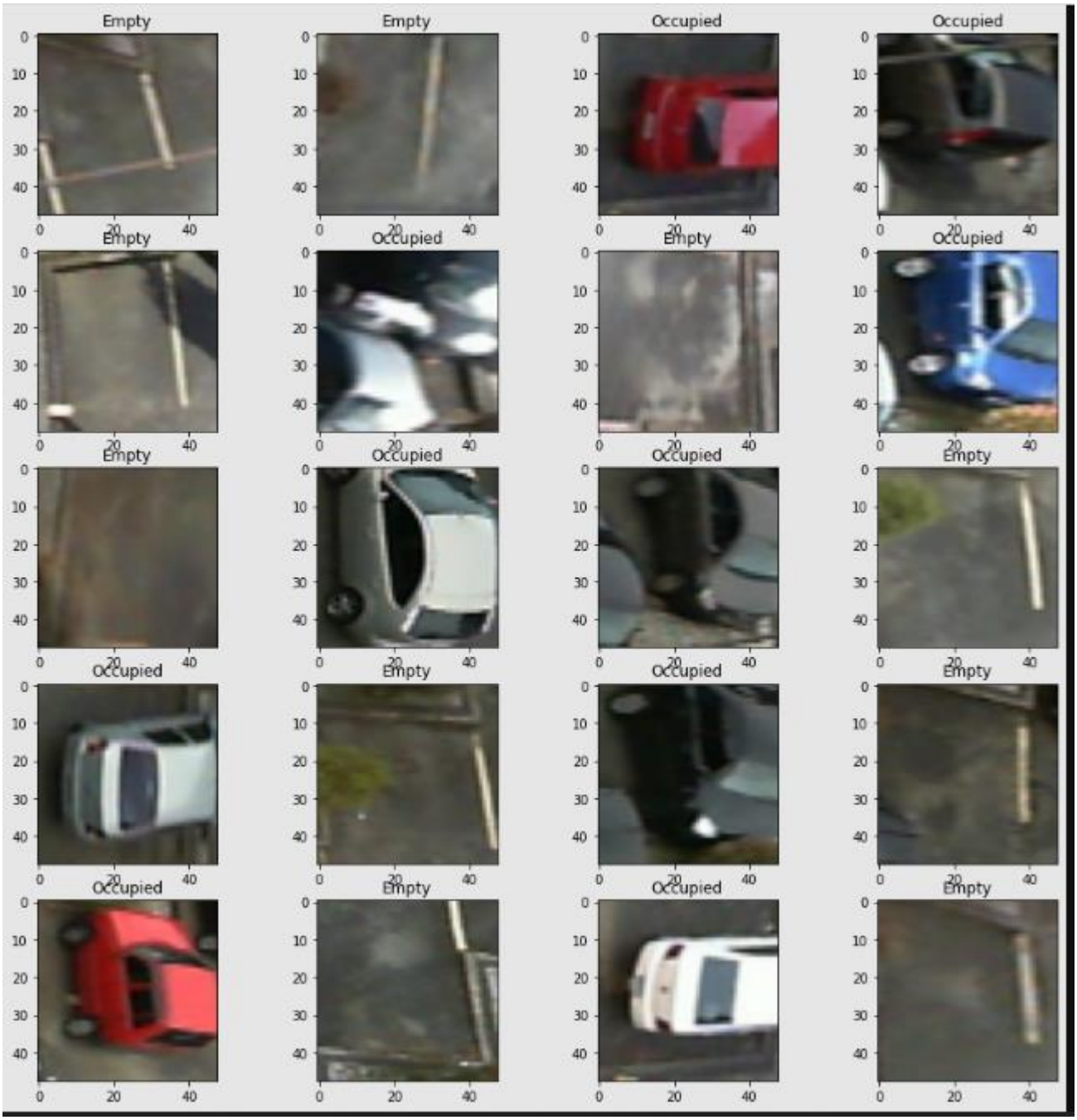

Figure 21: Random view of training images

c. **Data Pre-processing**

The data normalization (also referred to as data pre-processing) is a basic element of data mining. It means transforming the data, namely converting the source data in to another format that allows processing data effectively. It is the process of scaling individual samples to have unit norm (i.e from 0 to 1). Since we are working with images, the lowest pixel value is 0 and the highest is 255, we perform normalization on our image data using the numpy array library as shown in Figure 22

```
1 # Normalize pixel values to be between 0 and 1
2 train_images, test_images = train_images / 255.0, test_images / 255.0
```

Figure 22 Data Normalization

d. **Custom CNN Architecture**

With an image size of 48x48 as input, our CNN model takes tensors of shape (48, 48, 3). To complete the model, we feed the last output tensor from the convolutional base into flatten layer. This unrolls the 3D output to 1D, then we add our final Dense layers with the last layer (output layer) having a sigmoid activation function as seen in Figure 23. The sigmoid activation follows the sigmoid function as see in equation (3.4).

$$s(x) = \frac{1}{1+ e^{-x}} \qquad \textbf{(3.4)}$$

Sigmoid function

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 46, 46, 32)        896

 max_pooling2d (MaxPooling2D  (None, 23, 23, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 21, 21, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 10, 10, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 8, 8, 64)          36928

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 64)         0
 2D)

 conv2d_3 (Conv2D)           (None, 2, 2, 64)          36928

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 64)                16448

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 109,761
Trainable params: 109,761
Non-trainable params: 0
_________________________________________________________________
```

Figure 23 Custom CNN Architecture

After training our model for 10 epochs, we got a training accuracy of 99.9% and a validation accuracy of 99.89% as shown in Figure 24

```
Epoch 1/10
2625/2625 [==============================] - 330s 125ms/step - loss: 0.0372 - accuracy: 0.9883 - precision: 0.9862 - recall: 0.9904 - val_loss: 0.0106 - val_accuracy: 0.9970 - val_precision: 0.9965
Epoch 2/10
2625/2625 [==============================] - 321s 122ms/step - loss: 0.0116 - accuracy: 0.9968 - precision: 0.9968 - recall: 0.9967 - val_loss: 0.0122 - val_accuracy: 0.9969 - val_precision: 0.9996
Epoch 3/10
2625/2625 [==============================] - 320s 122ms/step - loss: 0.0078 - accuracy: 0.9982 - precision: 0.9984 - recall: 0.9979 - val_loss: 0.0192 - val_accuracy: 0.9942 - val_precision: 0.9995
Epoch 4/10
2625/2625 [==============================] - 318s 121ms/step - loss: 0.0065 - accuracy: 0.9983 - precision: 0.9987 - recall: 0.9979 - val_loss: 0.0058 - val_accuracy: 0.9989 - val_precision: 0.9990
Epoch 5/10
2625/2625 [==============================] - 316s 120ms/step - loss: 0.0054 - accuracy: 0.9986 - precision: 0.9989 - recall: 0.9982 - val_loss: 0.0071 - val_accuracy: 0.9986 - val_precision: 0.9986
Epoch 6/10
2625/2625 [==============================] - 314s 120ms/step - loss: 0.0046 - accuracy: 0.9988 - precision: 0.9990 - recall: 0.9985 - val_loss: 0.0057 - val_accuracy: 0.9990 - val_precision: 0.9994
Epoch 7/10
2625/2625 [==============================] - 316s 120ms/step - loss: 0.0047 - accuracy: 0.9987 - precision: 0.9990 - recall: 0.9984 - val_loss: 0.0075 - val_accuracy: 0.9989 - val_precision: 0.9991
Epoch 8/10
2625/2625 [==============================] - 314s 119ms/step - loss: 0.0051 - accuracy: 0.9988 - precision: 0.9990 - recall: 0.9985 - val_loss: 0.0060 - val_accuracy: 0.9989 - val_precision: 0.9987
Epoch 9/10
2625/2625 [==============================] - 312s 119ms/step - loss: 0.0035 - accuracy: 0.9990 - precision: 0.9993 - recall: 0.9986 - val_loss: 0.0074 - val_accuracy: 0.9984 - val_precision: 0.9992
Epoch 10/10
2625/2625 [==============================] - 310s 118ms/step - loss: 0.0040 - accuracy: 0.9990 - precision: 0.9994 - recall: 0.9987 - val_loss: 0.0068 - val_accuracy: 0.9989 - val_precision: 0.9990
```

Figure 24 Training after 10 epochs

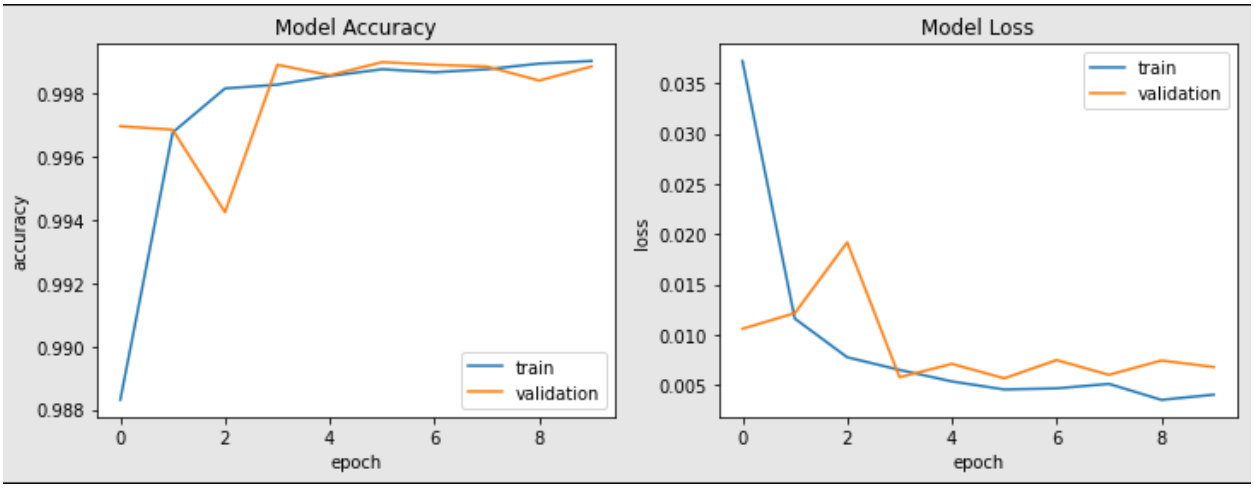

Figure 25 Model accuracy and loss plots for training and validation

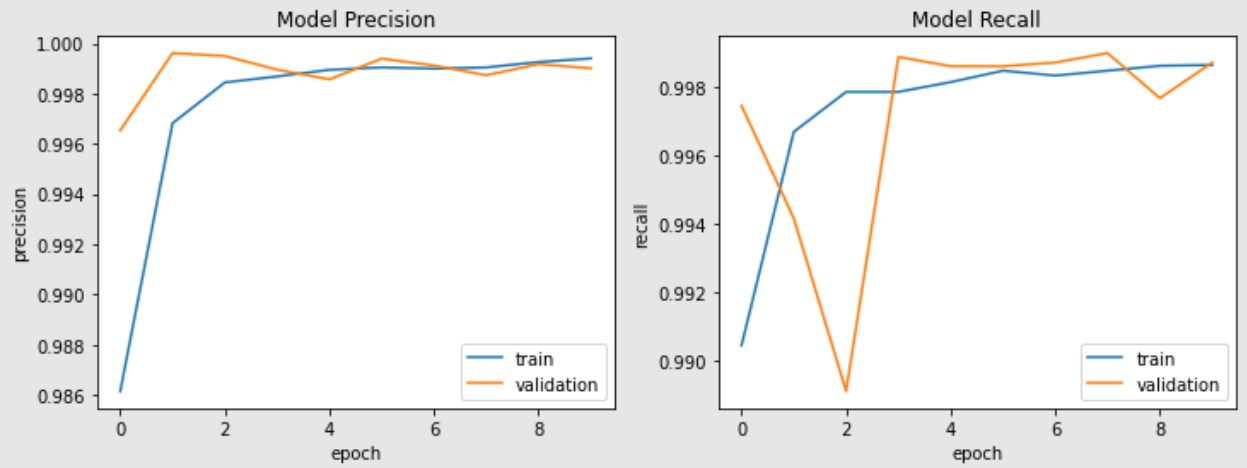

Figure 26 Model precision and recall plots for training and validation

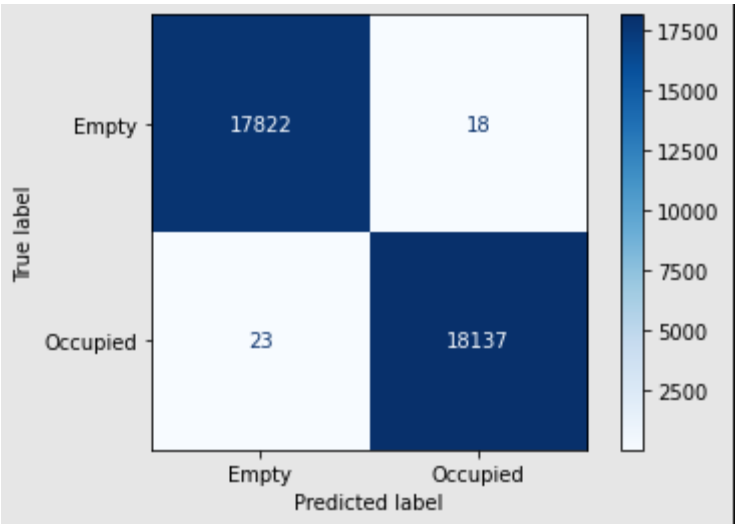

Figure 27 Confusion Matrix

We can see from Figure 25 and Figure 26 that our models performs well. Figure 27 shows a confusion matrix from which we observe that our model correctly predicts 17,822 slots as empty. This is called true positive (TP) and also predicts 18,137 occupied slots as occupied. This is known as true negative (TN). Alternatively, it predicted 23 occupied slots as empty (False Positive FP) and 18 empty slots as occupied (False Negative, FN). From these metrics, we can then compute our accuracy, precision and recall values from equations (3.3), (3.4) and (3.5) respectively.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \textbf{(3.5)}$$

$$Precision = \frac{TP}{TP+TN} \quad \textbf{(3.6)}$$

$$Recall = \frac{TN}{TP+FN} \quad \textbf{(3.7)}$$

Applying the above equations, we obtain an accuracy, precision and recall of 99.89%, 99.87% and 99.90% respectively.

e. Custom CNN Architecture (using Transfer Learning)

Using the same dataset as above, we created a CNN model leveraging on the VGG16 Net to perform binary classification as seen in Figure 28. In order to use the VGG16 Net base mode, we unfreeze the top layer and fused our layer having our desire output. We added two dense layer and finally the output layer with a sigmoid activation function.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 1, 1, 512)         14714688

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 50)                25650

 dense_1 (Dense)             (None, 20)                1020

 dense_2 (Dense)             (None, 1)                 21

=================================================================
Total params: 14,741,379
Trainable params: 26,691
Non-trainable params: 14,714,688
```

Figure 28 CNN Architecture using VGG16Net (Transfer Learning)

After running for 10 epochs, we obtain the following results.

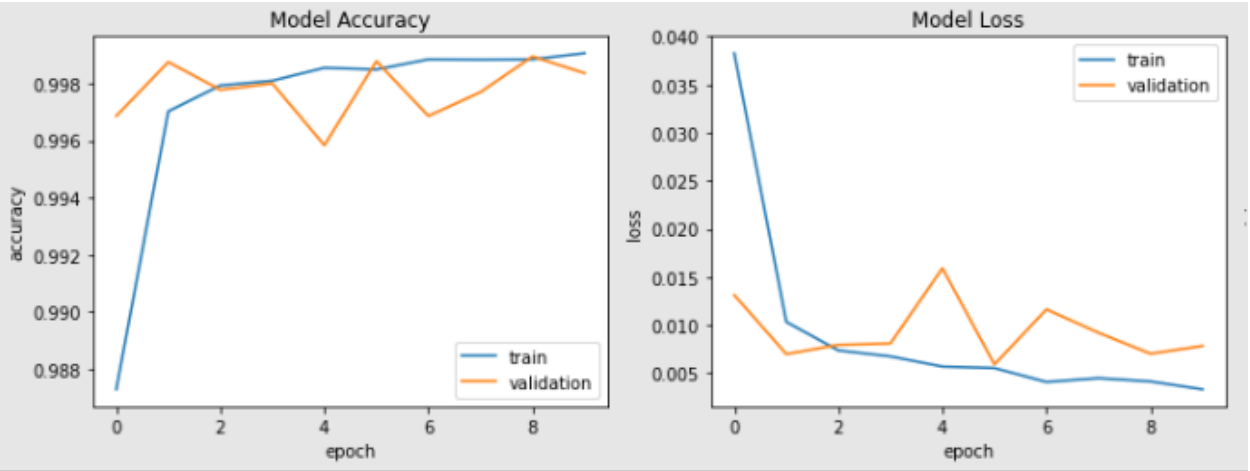

Figure 29 Training and Validation Accuracy, Loss Plots

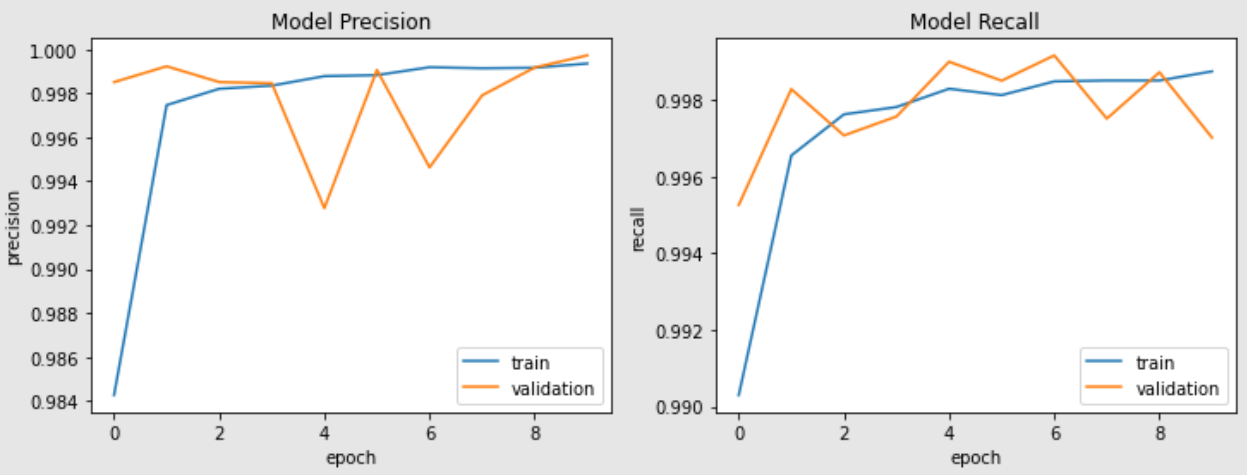

Figure 30 Training and Validation Precision, Loss Plots

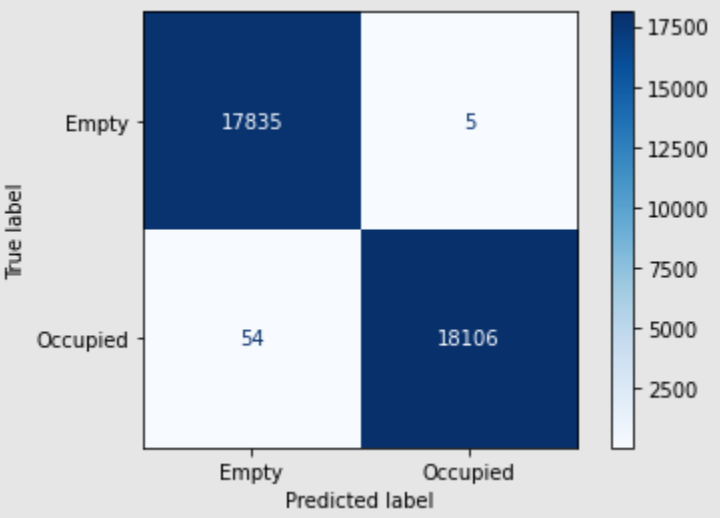

Figure 31 Confusion Matrix

From the Figure 31 above, we see that we obtained an accuracy of 99.84%, precision of 99.67% and a recall of 99.97

3.2.1. Parking Occupancy Detection Results

To detect parking occupancy, we run our CNN model through every slots of the current frame. This predicts the occupancy status of the slot as either empty or occupied as shown below.

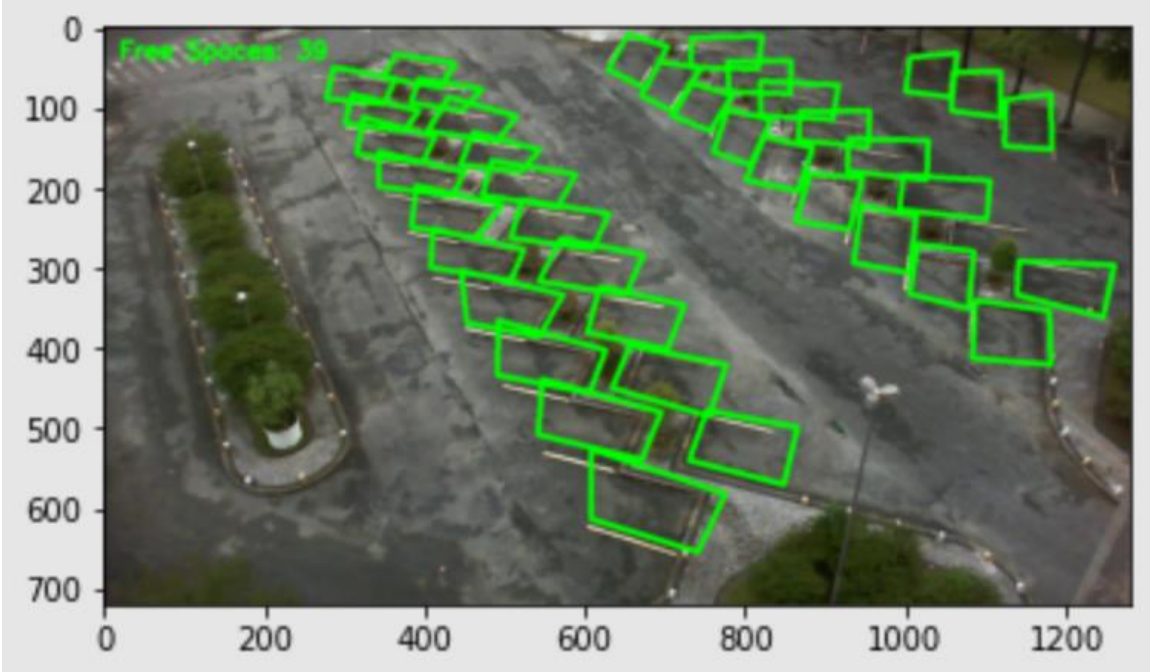

Figure 32 Empty Slots detection (Rainy Weather)

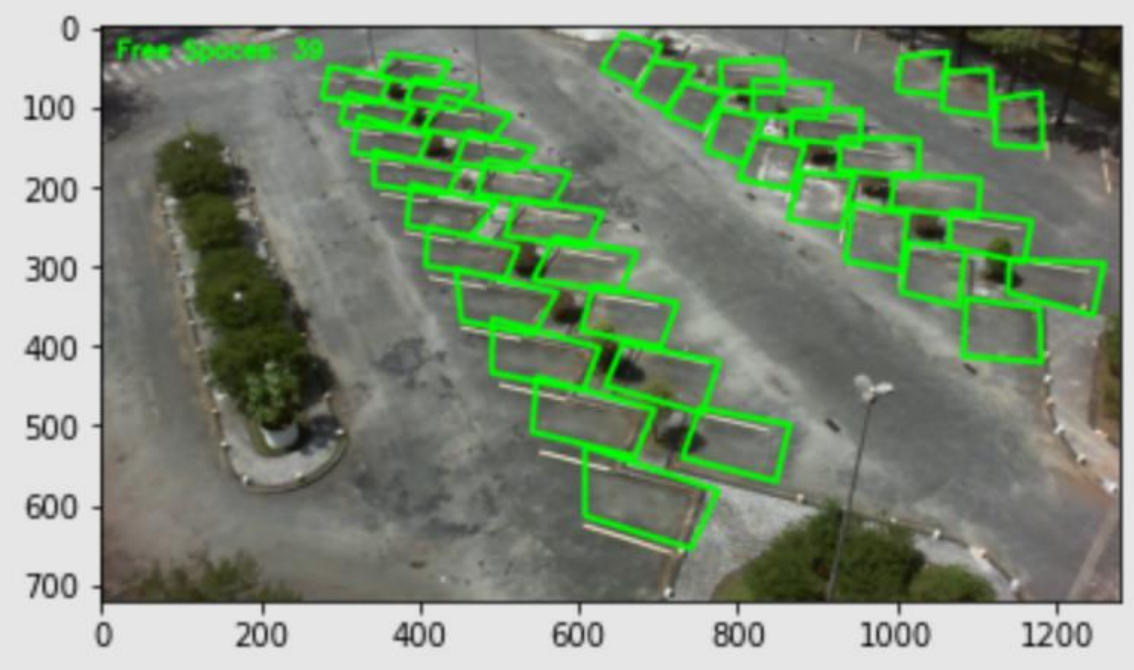

Figure 33 Empty Slots detection (Sunny Weather)

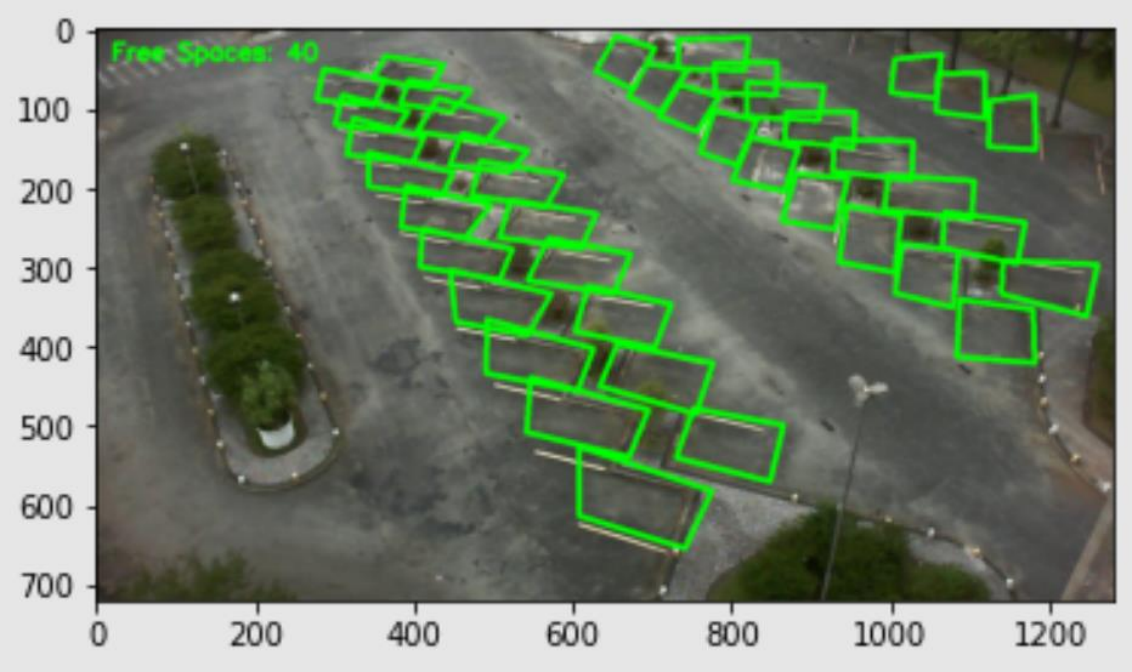

Figure 34 Empty Slots detection (Cloudy Weather)

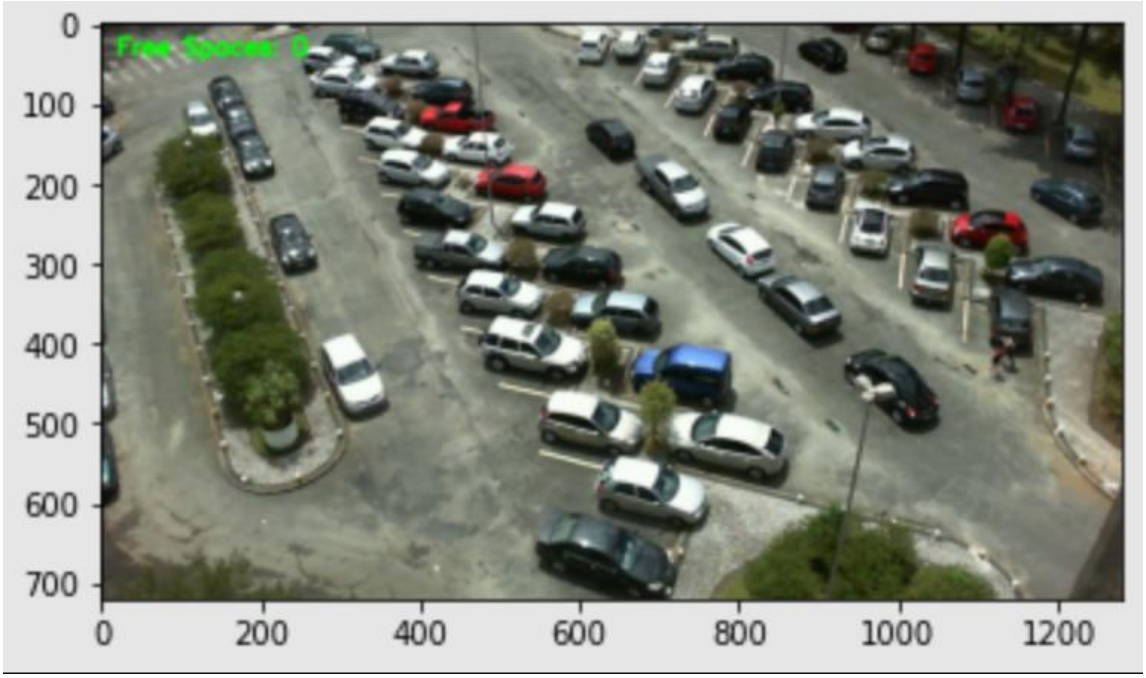

Figure 35 Parking Detection Result (Sunny Weather)

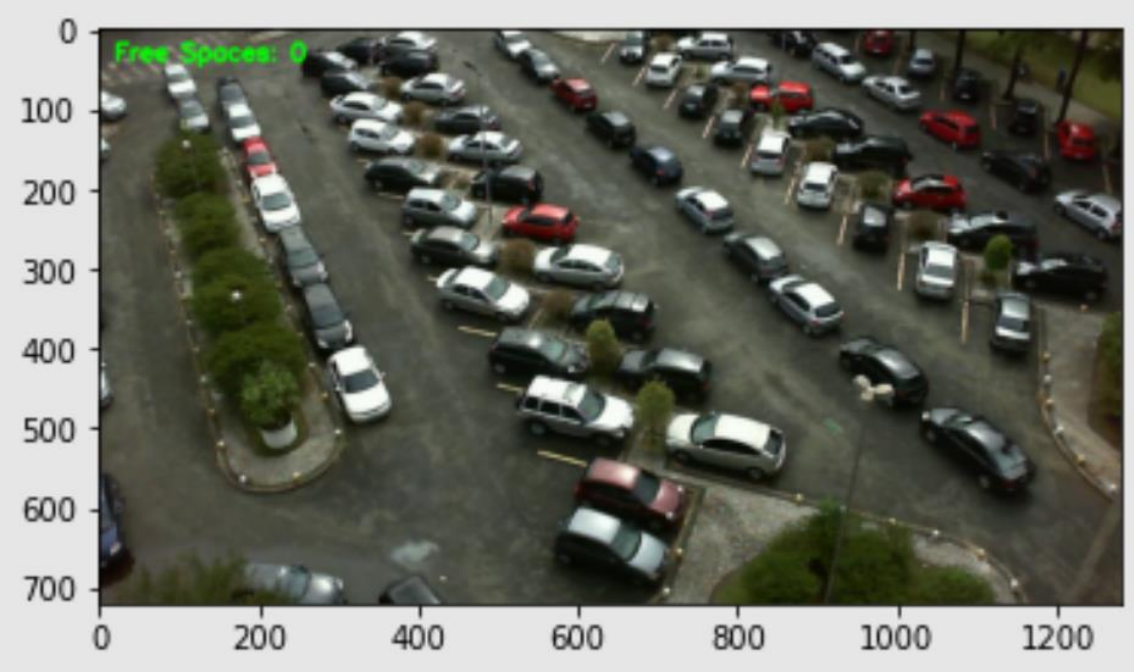

Figure 36 Parking Detection Result (Cloudy Weather)

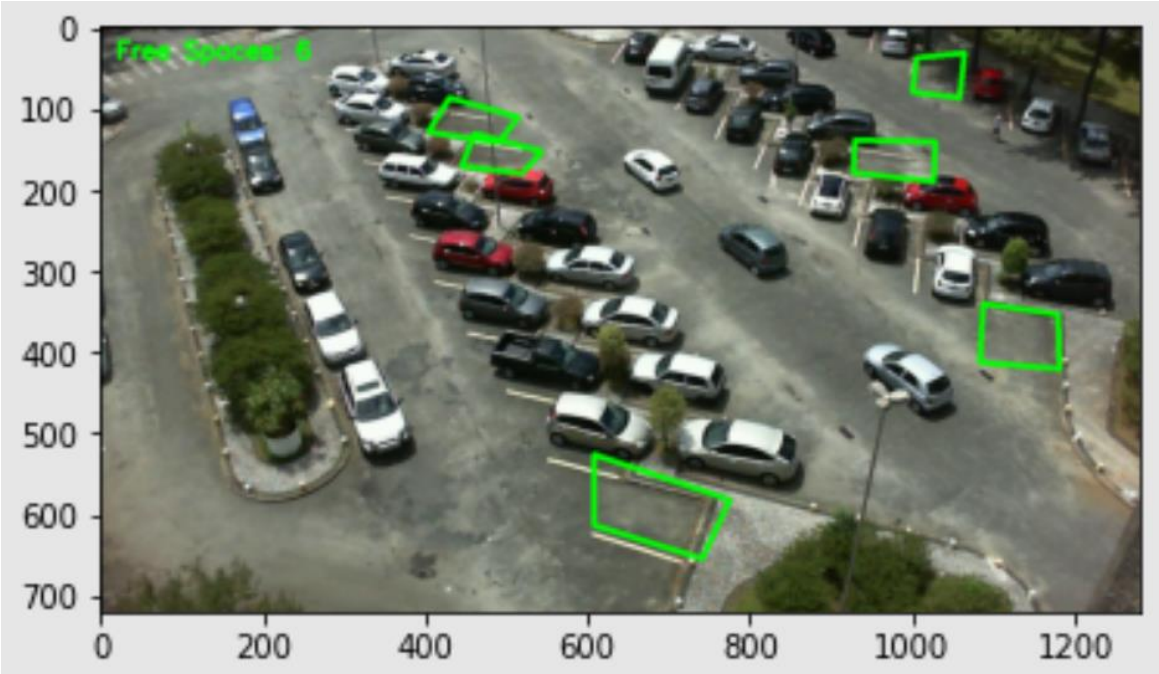

Figure 37 Parking Detection Result (Sunny Weather)

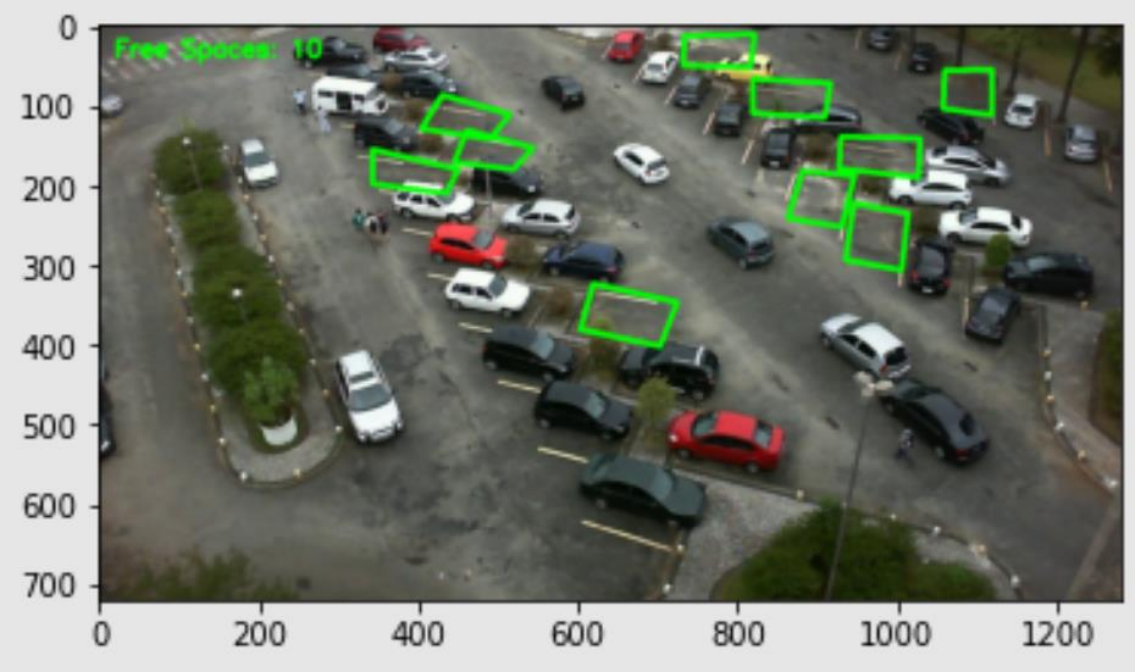

Figure 38 Parking Detection Result (Cloudy Weather)

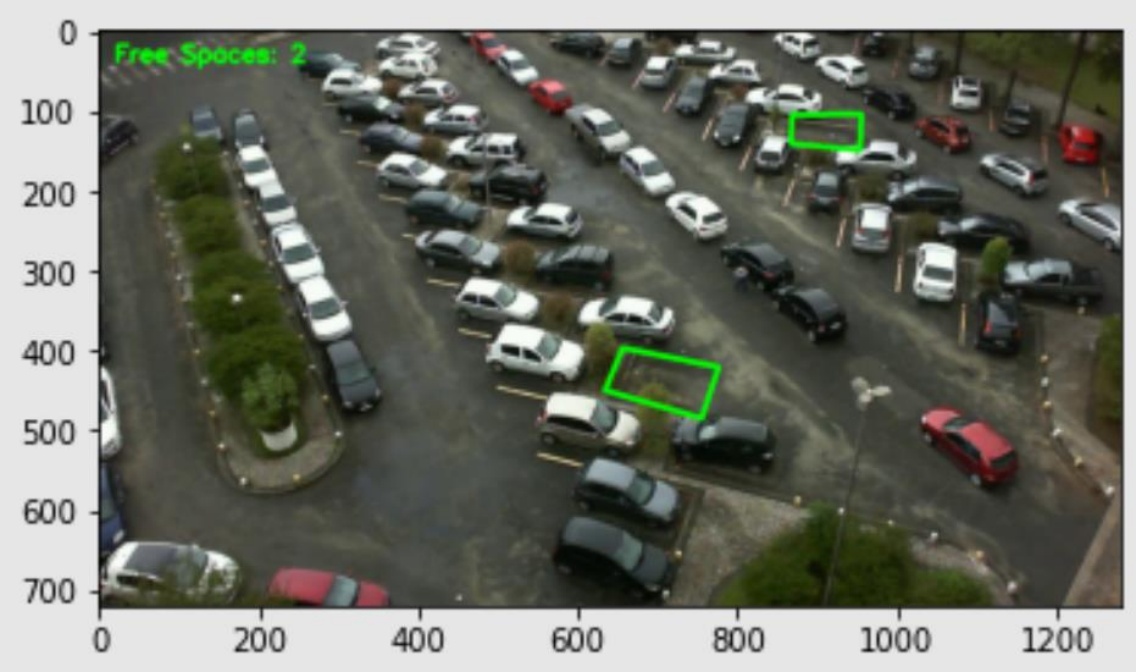

Figure 39 Parking Detection Result (Rainy Weather)

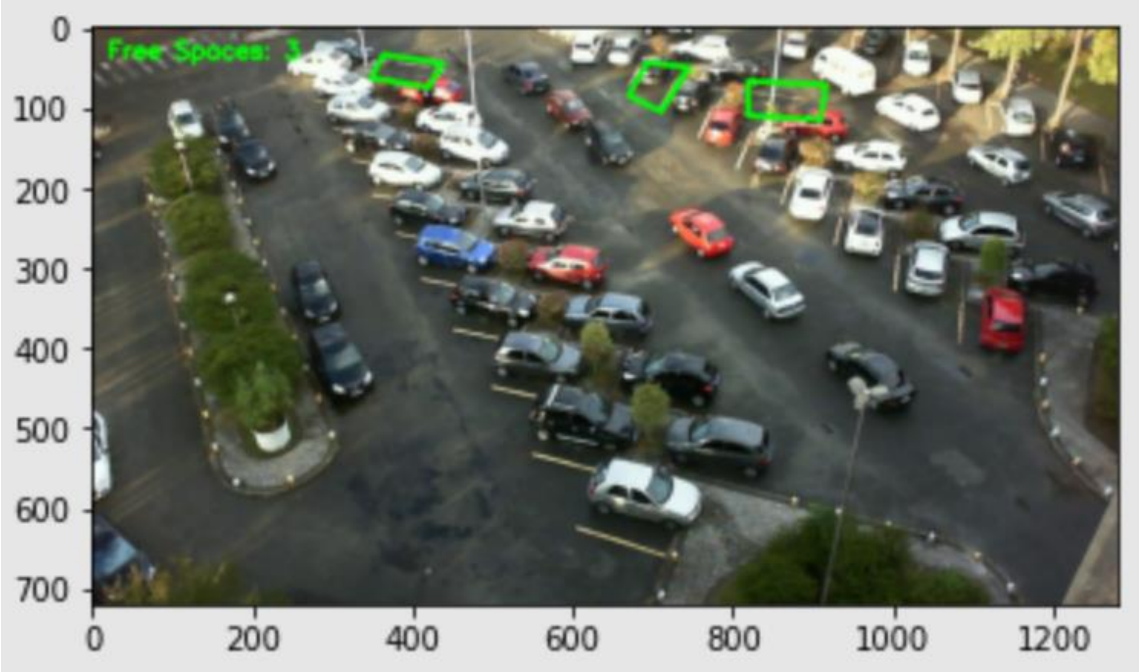

Figure 40 Parking Detection Result (Sunny Weather)

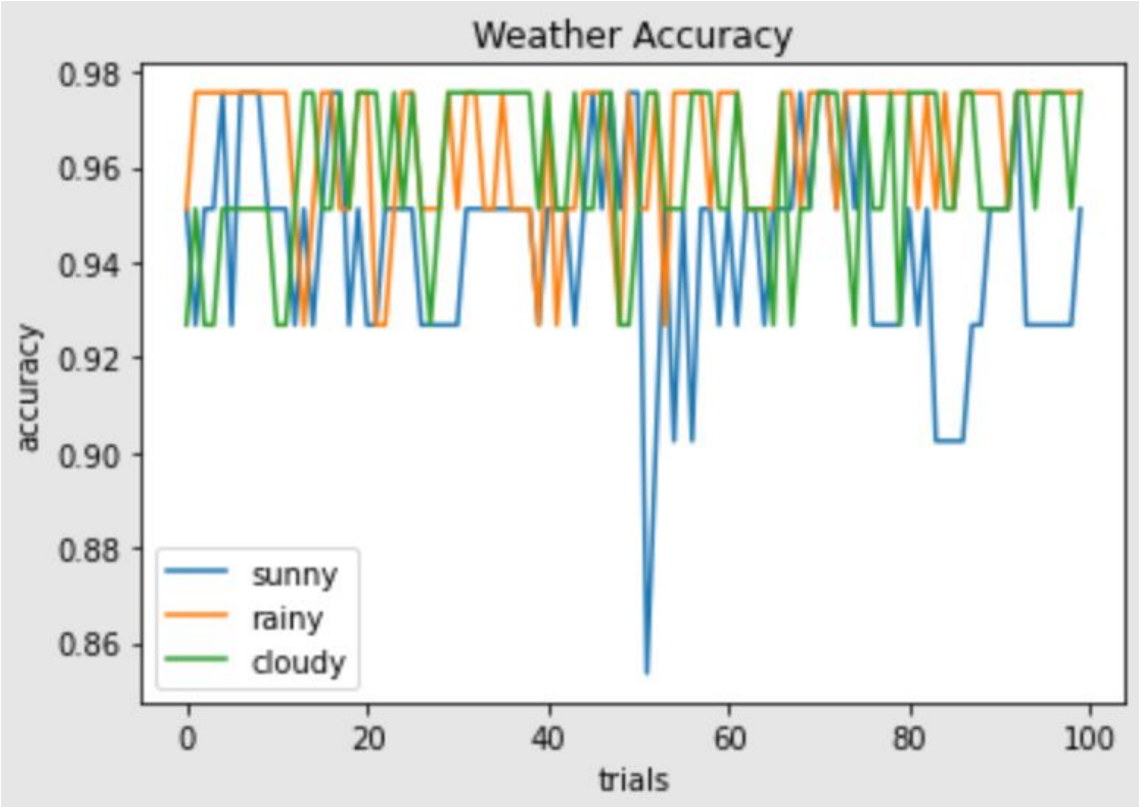

Figure 41 Parking Slot Detection Accuracy Chart

We ran our model through 100 samples each of rainy, cloudy and sunny weathers respectively. From Figure 41, we can see that our model performs quite well with the lowest accuracy of 85% experienced during the sunny period. This shows our model is quite robust for major weather conditions.

f. Custom CNN Architecture (using Transfer Learning for smaller dataset)

Transfer learning is useful to obtaining a more accurate result when we have less data. For this scenario, we extract images from a short clip of a parking lot (Neiezhmakov, 2019) and manually extracting our slot positions using the OpenCV library that allows us mark the spot on a frame with ease as shown in Figure 42. We classified those extract and grouped them into two classes (empty and occupied).

Figure 42 Marking of Parking Slots using python script

We have 632 training images which further splits into 314 for the occupied class and 318 for the empty class as we can see in Figure 43

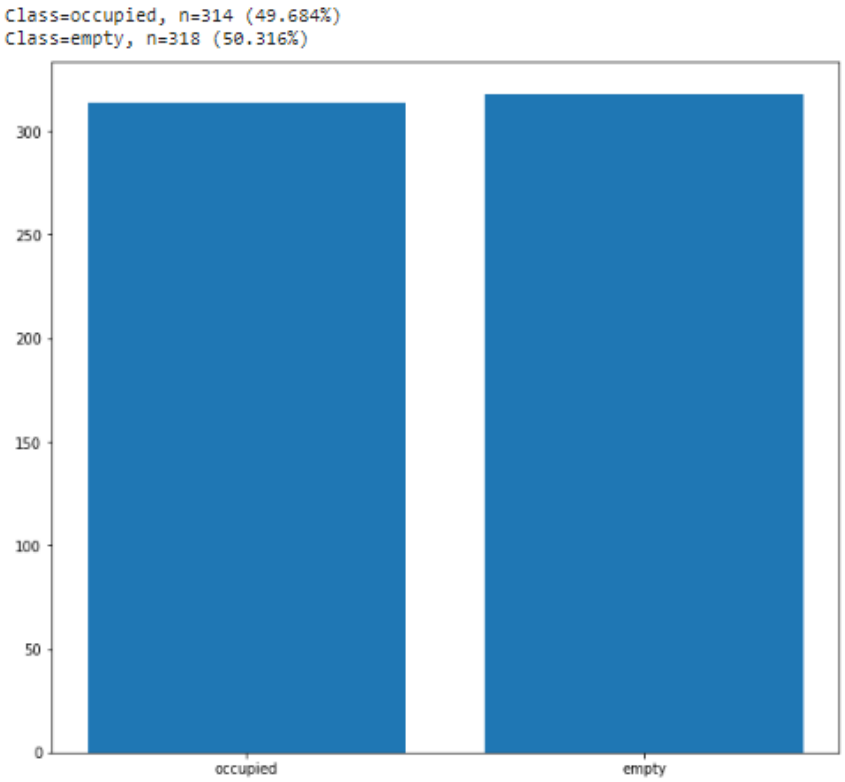

Figure 43 Class data distribution

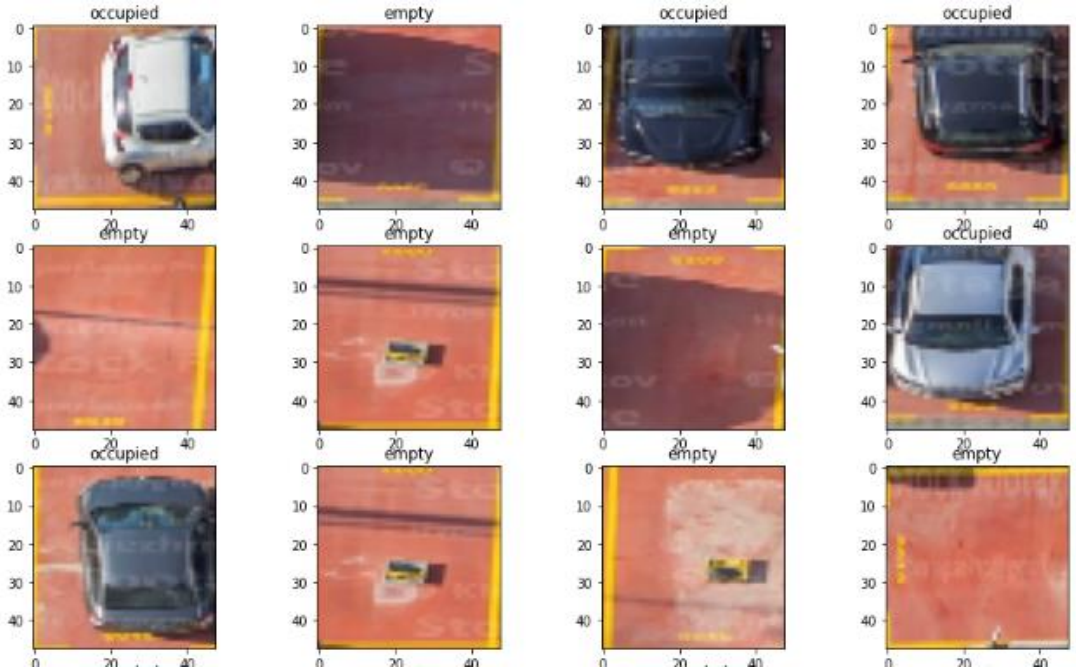

Figure 44 Extracted Slots

Using same transfer learning model shown in Figure 28, we see that we have about 14million parameters of which about 26 thousands are trainable. We see that our network is very deep yet we spend less time training it. This is the advantage to using transfer learning. Here, our model predicts if a marked slot is empty or occupied.

```
Epoch 1/10
20/20 [==============================] - 20s 918ms/step - loss: 0.3546 - accuracy: 0.9161 - precision: 0.9783 - recall: 0.8522 - val_loss: 0.1269 - val_accuracy: 0.9950 - val_precision: 0.9901
Epoch 2/10
20/20 [==============================] - 18s 890ms/step - loss: 0.0652 - accuracy: 1.0000 - precision: 1.0000 - recall: 1.0000 - val_loss: 0.0335 - val_accuracy: 1.0000 - val_precision: 1.0000
Epoch 3/10
20/20 [==============================] - 18s 889ms/step - loss: 0.0179 - accuracy: 1.0000 - precision: 1.0000 - recall: 1.0000 - val_loss: 0.0163 - val_accuracy: 1.0000 - val_precision: 1.0000
Epoch 4/10
20/20 [==============================] - 18s 887ms/step - loss: 0.0086 - accuracy: 1.0000 - precision: 1.0000 - recall: 1.0000 - val_loss: 0.0102 - val_accuracy: 1.0000 - val_precision: 1.0000
Epoch 5/10
20/20 [==============================] - 18s 885ms/step - loss: 0.0052 - accuracy: 1.0000 - precision: 1.0000 - recall: 1.0000 - val_loss: 0.0075 - val_accuracy: 1.0000 - val_precision: 1.0000
Epoch 6/10
20/20 [==============================] - 18s 887ms/step - loss: 0.0037 - accuracy: 1.0000 - precision: 1.0000 - recall: 1.0000 - val_loss: 0.0056 - val_accuracy: 1.0000 - val_precision: 1.0000
Epoch 7/10
20/20 [==============================] - 18s 895ms/step - loss: 0.0026 - accuracy: 1.0000 - precision: 1.0000 - recall: 1.0000 - val_loss: 0.0043 - val_accuracy: 1.0000 - val_precision: 1.0000
```

Figure 45 Training after 7 epochs

From Figure 45, Our model was supposed to train for 10 epochs but it stopped at 7 due to the early stopping callback we implemented. Early stopping refers stopping the training process before the learner passes that point. (Jain and Sharma, 2020)

The extracted slots are then passed through the model for training and validation. Figure 46 and Figure 4711 shows model training and validation accuracy, loss, precision plots respectively.

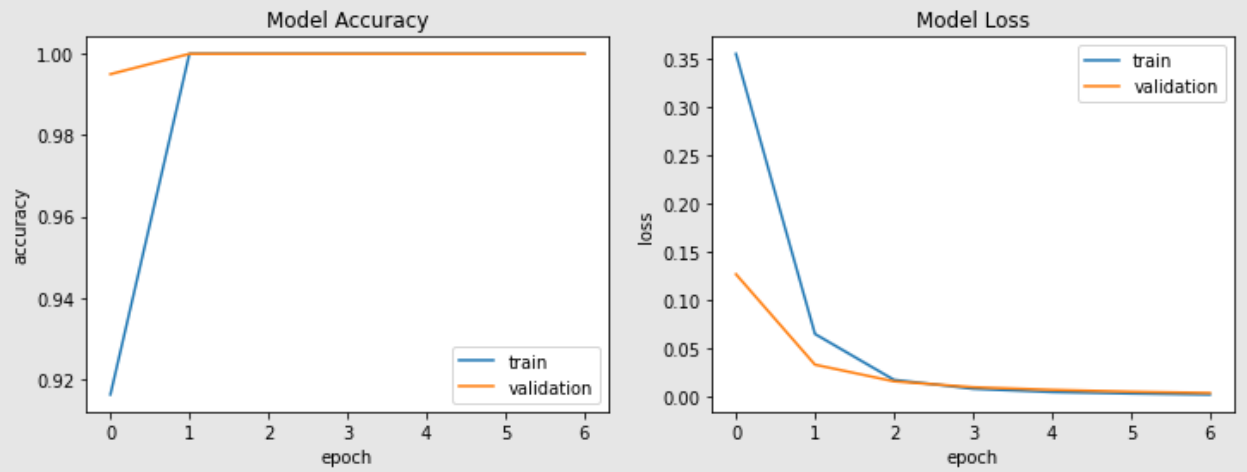

Figure 46 Model accuracy and loss plots for training and validation (transfer learning)

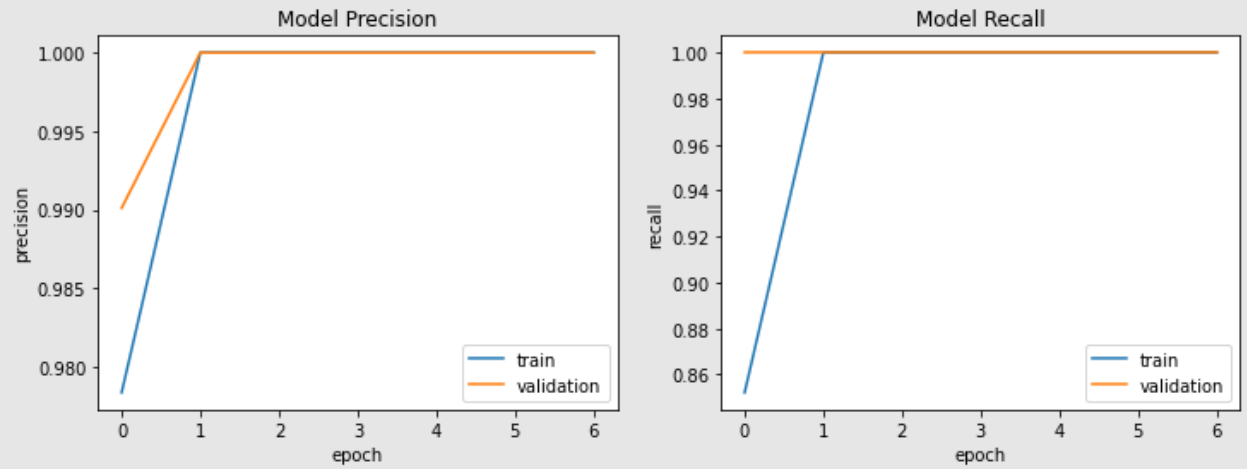

Figure 47 Model precision and recall plots for training and validation (transfer learning)

Figure 48 Model confusion matrix (transfer learning)

Figure 48 shows a confusion matrix from which we observe that our model correctly predicts 100 empty slots as empty and 100 occupied slots as occupied.

Figure 49 Parking Occupancy Detection Result

Figure 50: Parking Occupancy Detection Result

The idea behind using the transfer learning technique is to prepare for special situations where our general model performs badly and quick deployment is imminent. We can see

from Figure 45 that the total training time was 128 seconds. Using our slot marking script and transfer learning together makes a framework that adapts to different parking structures and weather.

CHAPTER FOUR

4. CONCLUSION

This research developed a system for detecting parking space occupancy using machine learning through video data analysis from the outdoor parking area. We proposed two main approaches. The first was using Mask RCNN. This method is suitable for a camera positioned at an angle. This involved two main stages; the first was identifying the region of interest by marking the parking position on the image of a full parking lot. The second stage is feature extraction based on the region of interest (ROI) of each slot in the parking area. Experiments for parking space detection are evaluated using the accuracy of IoU. The second proposed idea was to use a custom trained CNN to predict the status of a parking spot. This method works best for cameras positioned directly above the parking lot. Our CNN, in particular, exhibits very good accuracy even in the presence of noise caused by light condition changes, shadows, and partial occlusions under various weather conditions, and its implementation provides accurate automatic parking space occupancy detection while reducing unnecessary maintenance installation costs.

Further work can be carried out to include other weather conditions like snow, harmattan, wind, etc. This will make the model robust and more efficient. However, in such a scenario, using our marking script and custom transfer learning implementation may prove sufficient. Auto detecting the region of interest and parking slots can be further researched as this will vital to creating a solution robust for all parking lots. For commercial deployment, I would strongly recommend the use of a web and mobile framework as this will largely spread the impact of this solution to reduce congestion faced due to a lack of available parking slots.

.

REFERENCES

Almeida, P. *et al.* (2015) 'PKLot - A Robust Dataset for Parking Lot Classification', *Expert Systems with Applications*, 42. doi:10.1016/j.eswa.2015.02.009.

Amato, G. *et al.* (2017a) 'Deep learning for decentralized parking lot occupancy detection', *Expert Systems with Applications*, 72, pp. 327–334. doi:10.1016/j.eswa.2016.10.055.

Amato, G. *et al.* (2017b) 'Deep learning for decentralized parking lot occupancy detection', *Expert Systems with Applications*, 72, pp. 327–334. doi:10.1016/j.eswa.2016.10.055.

Chollet, F. (2015) *Keras*. Available at: https://keras.io/about/.

Evermann, J., Rehse, J.-R. and Fettke, P. (2017) 'XES tensorflow-Process prediction using the tensorflow deep-learning framework', *arXiv preprint arXiv:1705.01507* [Preprint].

He, K. *et al.* (2017) 'Mask R-CNN', in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988. doi:10.1109/ICCV.2017.322.

Hunter, J.D. (2007) 'Matplotlib: A 2D graphics environment', *Computing in science & engineering*, 9(03), pp. 90–95.

Jain, A.K. and Sharma, M.P. (2020) 'MODELING ERROR', *Journal of Analysis and Computation (JAC) (An International Peer Reviewed Journal), www.ijaconline.com, ISSN 0973-2861*, XIV(VI).

Kumar, A. *et al.* (2021) 'A drone-based networked system and methods for combating coronavirus disease (COVID-19) pandemic', *Future Generation Computer Systems*, 115, pp. 1–19. doi:10.1016/j.future.2020.08.046.

Lee, S., Yoon, D. and Ghosh, A. (2008) 'Intelligent parking lot application using wireless sensor networks', in *2008 International Symposium on Collaborative Technologies and Systems*. IEEE, pp. 48–57.

Naufal, A., Fatichah, C. and Suciati, N. (2020) 'Preprocessed Mask RCNN for Parking Space Detection in Smart Parking Systems', *International Journal of Intelligent*

Engineering and Systems, 13(6), pp. 255–265. doi:10.22266/ijies2020.1231.23.

Neiezhmakov, K. (2019) *Car parking lot viewed from above timelapse, Aerial view*. Available at: https://www.youtube.com/watch?v=Pv8N1PamwPQ (Accessed: 28 April 2022).

Oliphant, T.E. (2006) *A guide to NumPy*. Trelgol Publishing USA.

Shao, Y., Chen, P. and Cao, T. (2018) 'A Grid Projection Method Based on Ultrasonic Sensor for Parking Space Detection', in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, pp. 3378–3381. doi:10.1109/IGARSS.2018.8519022.

Simonyan, K. and Zisserman, A. (2015) 'Very deep convolutional networks for large-scale image recognition', *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–14.

Tanuwijaya, E. and Fatichah, C. (2020) 'MODIFICATION OF ALEXNET ARCHITECTURE FOR DETECTION OF CAR PARKING AVAILABILITY IN VIDEO CCTV', *Jurnal Ilmu Komputer dan Informasi*, 13(2), pp. 47–55. doi:10.21609/jiki.v13i2.808.

Thomas, T. and Bhatt, T. (2018) 'Smart Car Parking System Using Convolutional Neural Network', in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*. IEEE, pp. 172–174. doi:10.1109/ICIRCA.2018.8597227.

Vuk, D. and Andročec, D. (2022) 'Application of Machine Learning Methods on IoT Parking Sensors' Data BT - Proceedings of Sixth International Congress on Information and Communication Technology', in Yang, X.-S. et al. (eds). Singapore: Springer Singapore, pp. 157–164.

Wu, T., Zhao, H. and Ou, X. (2014) 'Vehicle Ownership Analysis Based on GDP per Capita in China: 1963–2050', *Sustainability*, 6(8), pp. 4877–4899. doi:10.3390/su6084877.