Evolution of Information, Communication and Computing Systems (EICCS)
https://publisher.uthm.edu.my/bookseries/index.php/eiccs

IoT based Parking Management System using Video Processing for Open Space Parking Area

Saravanaraj Sathasivam[1], Abd Kadir Mahamad[1*], Sharifah Saon[1], Sri Wiwoho Mudjanarko[2], Shingo Yamaguchi[3], and Mohd Anuaruddin Ahmadon[3]

[1]Faculty of Electrical and Electronic Engineering,
University Tun Hussien Onn Malaysia, MALAYSIA

[2]Civil Engineering,
Universitas Narotama, Surabaya 60117, INDONESIA

[3]Graduate School of Science and Technology for Innovation,
Yamaguchi University, JAPAN

*Corresponding Author

Abstract:
The flawed parking management system may waste time and cost searching for parking spaces. It may also cause traffic congestion and carbon emission. Finding a parking space in most metropolitan areas is difficult for drivers, especially during rush hours. A difficulty arises from not knowing where the available spaces may be when traffic congestion may occur. The proposed system avoids poor parking management through object recognition technology detecting parking spaces of modern busy parking lots. Traffic congestion reduces by notifying the driver exact vacant parking spaces location. So, the users can park their car in available space directly without looking around for space. The main objective of this project is to design IoT based parking system using TensorFlow based on object detection technology, followed by the integration of the developed IoT parking system for normal road users. The project used Anaconda prompt to run the TensorFlow main object detection code. The Telegram app acts as an IoT platform for users who can notify about the available or non-available parking. For the demo purpose, the project uses a video feed as the input for object detection. As a result, the parking status data counts on the terminal for object detections in Anaconda prompt. At the same time, Telegram sends the notification to the user. In conclusion, the system proposes a practical methodology for managing the parking system using a camera placed for a lamp-post view. The system's accuracy depends on the training model used for object training and the number of samples provided for training.

Keywords:
IoT · Parking management system · Video processing · Tensorflow · Telegram Bot app

1. Introduction

Nowadays, many existing car parks do not have a systematic system. Most of the existing car parking systems are operated manually and are inefficient. The problem at the car park is time wasted searching for the available parking spaces. Users will circle the parking area until they find a vacant parking slot

*Corresponding author: kadir@uthm.edu.my

[1]. In urban areas, where the number of cars is greater than the availability of parking spaces, this issue mainly occurs. These inefficient situations have arisen caused by the lack of adoption of the technologies available on the market today. Most of the existing parking system is designed for indoor parking, less parking system for open area parking [2].

So, this system is designed mainly for open area parking. This system is proposed, classifying the parking space as occupied or vacant. Such a system is beneficial when a camera is placed at a lamp post view and visible parking spaces. The proposed system is used just a camera that has been pre-programmed to detect the parking lot vehicle. This system could work with minimum hardware and software requirements such as Python Anaconda Prompt, Telegram Bot, and TensorFlow packages. In addition, this system doesn't need to have any circuit or sensor to operate this system. This system proposed to use object detection for object recognition and motion tracking to distinguish between the parked and vacant spaces [3]. Python is the best platform for object detection and motion tracking using the TensorFlow SSD Mobilenet model. Python supports object detection and motion tracking libraries available in Anaconda Navigator. The count of each available space and non-available space from the output terminal of Anaconda Prompt can push data to Telegram Server, which can be retrieved back in a Telegram App.

2. Literature Review

Poor parking management influences driver's search time and cost for parking spaces. It may also cause traffic congestion [4]. Finding a parking space in most metropolitan areas is difficult for drivers, especially during rush hours. A difficulty arises from not knowing where the available spaces may be when traffic congestion may occur. In order to increase the number of vehicles and the decreasing efficiency of modern busy parking lots, the proposed system avoids poor parking management through object recognition technology that detects vacant spaces and parking spaces. Vehicle parking can be categorised into indoor and open space parking (outdoor). Indoor parking allows you to park your automobile in a generally secure, covered area, away from potential weather or criminal damage. Outdoor parking provides a secure location that is normally not accessible to the general public and is susceptible to weather conditions.

Traffic congestion reduces by notifying the driver exact vacant parking spaces location. So, the users can park their car in available space directly without looking around for empty space. The Telegram Server collected data should be retrieved and displayed in Telegram Bot App to help users find parking spaces from anywhere. This system proposed to use Telegram Bot App for user operation to friendly user interface purposes [5].

The object detection system consists of various types of methods. The following part discusses the findings of some reference books and research papers based on the various types of object detection technology using Python. This part is able to understand readers about the existing object detection technology and its advantages and disadvantages. Object Detection technology is frequently used in a variety of applications such as vehicle detection, face recognition and detection, autonomous vehicles and pedestrians on streets. Krishna et al. used TensorFlow Object Detection API, which is a tool for object detection methods [6]. TensorFlow is the latest tool for object detection that allows anyone to design and deploy an efficient image recognition model. Object detection involves classifying and detecting objects in an image, locating them, and attracting bounding boxes. For project implementation, this paper used the Faster R-CNN method [6]. Threatening Objects are divided into two categories in the learned model. Two types of data are used to test the trained model.

R. Phadnis et al. in 2018 research on the object detection of objects available in a household [7]. The object detection was trained in terms of time and geographic movement. The research was designed to identify the things of households. The trained model keeps better track of the objects and maximises efficiency by reducing time wasted in forgetting or finding the object. This project also implemented TensorFlow recent new library packages from Google to train the model. The TensorFlow API is implemented to recognise multiple objects in real-time video streams. This project also introduces an algorithm to identify and detect patterns and alert the user [7].

Current advancements have substituted traditional machine learning approaches for identifying traffic signals and categorisation in deep learning object recognition methods, which have been made possible by developing convolution neural networks (CNN). The research papers implement a deep learning technique for powerful detection of light by comparing various object detection models and analysing the TensorFlow model's efficiency [8]. T. V. Janahiraman et al. analyse the advantages and disadvantages of Single Shot Detector (SSD) MobileNet V2 and Faster-RCNN [8]. The study shows Faster-RCNN delivers 97.01%, and Single Shot (SSD) delivers 38.80% for a model trained using 441 images. Meanwhile, D. Cao introduces object detection methods on Convolution Neural Network (CNN) in smoke detection [9]. This paper emphasises the smoke detector for an alarm system [9]. The research was carried out in various neural networks for object detector models such as Faster R-CNN, Single Shot Detector (SSD), and Region-based Fully Convolutional Network (R-CNN).

This study focuses on YOLO-LITE, a real-time object detection model designed to run on portable devices without a graphics processing unit, such as a laptop or mobile (GPU) [10]. The model was trained on the PASCAL VOC dataset before being applied to the COCO dataset by reaching mAP of 33.81% and 12.26%, respectively. YOLO-LITE works at about 21 FPS on the non-GPU computer and 10 FPS on a website. YOLO-LITE was created based on the original YOLOV2 object detection algorithm to produce a smaller, faster, and more efficient model, allowing real-time object detection to be accessible to a wider range of devices. YOLO has a quick detection speed and is appropriate for target identification in a real-time context as a target detection system. It boasts a higher detection accuracy and a faster detection time when compared to other similar target detection systems. Face detection is the subject of this study, which is based on the YOLO network. The YOLO target detection technique is used to detect faces in this article [11]. According to the experiments, the face. The detection approach based on YOLO has a higher resilience and detection speed. Even in a complex setting, good detection accuracy can be guaranteed. Detection speed can match real-time detection requirements at the same time.

3. Research Methodology

The Python code in this proposed system is to detect motion tracking and object recognition by threshold measurement. If the possibility of the present vehicle is beyond the threshold, the classifier is applied to detect the movement of vehicles. Thus, this system proposes a combination of detection, tracking and classification to achieve efficient parking. The Python code process the recorded video captured from the camera top view, and the output can be used for real-time application. The development of this video processing-based parking management system is separated into three phases, as in Fig. 1.

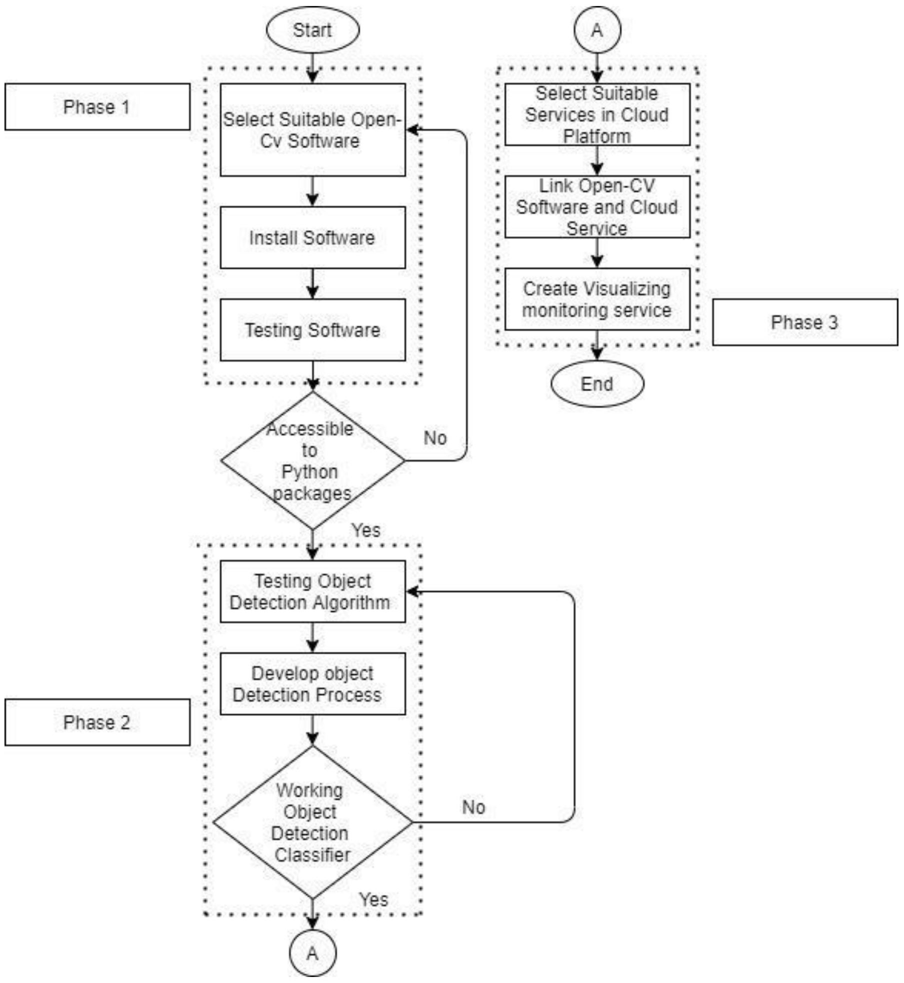

Start
Phase 1
Select Suitable Open-Cv Software
Install Software
Testing Software
Accessible to Python packages
No
Yes
Testing Object Detection Algorithm
Develop object Detection Process
Phase 2
Working Object Detection Classifier
No
Yes
A
A
Select Suitable Services in Cloud Platform
Link Open-CV Software and Cloud Service
Create Visualizing monitoring service
Phase 3
End

Figure 1: Flowchart overview of project development

3.1 Phase 1: Software for Parking Management System

This study describes the software being used to run the parking management system. Each software application's description and vital details were explained in this report. Python packages for this project have access to all of this stuff. Anaconda Navigator and C++ Notepad Editor are the main pieces of software required by this system.

3.2 Phase 2: Development of Object Detection using TensorFlow for Image Detection, Video Detection and Real-Time Detection

Step 1: Capture images of the parking top view
TensorFlow object detection models required a minimum of 300 pictures of samples. The sample pictures need to be separated into two category folders: Test folder and Train folder. The test folder must contain 20 % of parking pictures as a requirement for object detection. At the same time, the train folder needs to save 80 % of parking pictures.

Step 2: Labelling Python images for XML file conversion
After running the 'python labellmg.py' file into the Anaconda prompt, a labelling prompt appears. The 'labellmg.py' window prompt is required to set the directory of saved Test folder images and Train folder images. Then the output directory also needs to be set to the same directory as the Test folder and Train folder so the XML file can be saved in the same directory. The labellmg is a graphical image

annotation tool. The graphical annotation tool is created in Python. These annotations are saved as XML files in PASCAL VOC format.

Step 3: Convert the xml file into Excel csv file
When run the Python code, two excel file been created, which are for test labels and train labels. Inside Excel file of train labels and test labels have data about each image annotations such as filename, width, height, class, xmin, ymin, xmax, and ymax. Based on this research for parking detection consist of two type classes which are empty and occupied. The following method generates the data in Excel file into the record file to proceed for the following method to freeze the recorded data.

Step 4: Train recorded data into TensorFlow model (Training model)
The TensorFlow object detection API is the framework for creating a deep learning network that solves object detection problems. TensorFlow provides pre-trained models in their framework, which they refer to as Model Zoo. It includes pre-trained models trained on the COCO dataset, the KITTI dataset, and the Open Images Dataset. The selection of perfect models of TensorFlow based on the object detection that needed for a particular project. This research used MobileNet_SSD architecture to train the object detection for the parking system. The SSD architecture is a single convolution network that learns to predict bounding box locations and classify these locations in one pass.

Step 5: Export inference graph
When training begins, the TensorFlow started stepping through training batches and reporting the loss at each step. The lost value starting of the model training get high and then gets lower and lower as the object detection classifier trains. During the training process, the lost value should maintain below 0.05 to stop the training. In this process, the progress of the training can view by using TensorBoard. After the training, the highest checkpoint file should be stored in the training folder. The checkpoint file required to generate the frozen inference graph.

3.3 Phase 3: Development of Cloud Computing in Telegram Bot

A Telegram Bot is created to access token API that is included in the main code to transfer data in the Anaconda terminal to the Telegram Bot group. Therefore, the parking user receives a notification about the parking status. The following is a method to create Telegram Bot.

Step 1: Create Bot Father
Telegram Bot can be created in BotFather chatbot, which is essentially a bot used to create other bots. A command needs to be typed as /newbot, which takes to the next step for the bot creating process.

Step 2: Username Creation
The next step would be username creating process. The username is needed for the purpose of mentions and sharing. The bot username is required to be take noted for sending the data to telegram on the valid pathway. Sometimes the username can be found in the setting.

Step 3: Access Token Creation
After choosing a suitable username, the BotFather user gets an access token. The access token is advisable to save the access token and username for further programming steps. The username and token access are required to insert into the main programming code.

4. Result

4.1 The Outcome Result for Phase 2

The following part is the result and outcome of the object detection using TensorFlow 1.15 version. It has described the before and after process during object detection. There is some instruction to run object detection. It is first required to edit the main code in line 37 video _name path with the designated video to be tested. Then open the Anaconda Prompt, change the directory to the main code located and run it.

4.1.1 The Video Before the Object Detection Process

Fig. 2 shows one of the frames the video feed is taken from the Batu Pahat Street parking area. Packages needed to run video detection program. The main code is required to set the path for video detection, as mentioned before. If the chances of detecting a vehicle are higher than the threshold, then a classifier is applied to detect if it is parking available or not. If the TensorFlow detection detects the object as trained earlier, it will appear to mask whether empty or occupied in the video itself. If the motion is detected at the marked landmark, the classifier is called to calculate and determine vehicle occupancy.

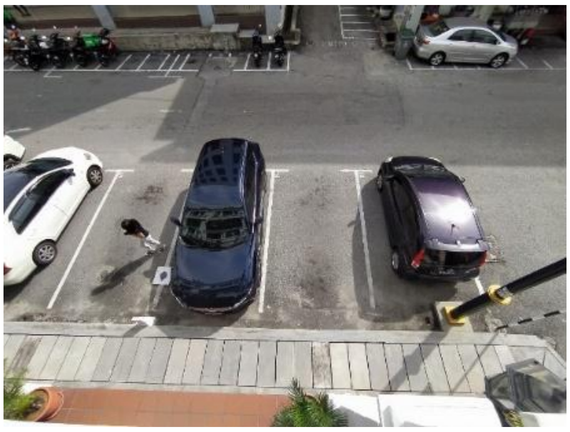


Figure 2: Video before Tensorflow object detection classifier process

4.1.2 The Video After the Object Detection Process

The result of the Anaconda prompt video object detection processing can be seen in Fig. 3. For this stage, the detection of a vehicle in each lot is important to ensure that the algorithm code detects parking availability without getting errors. The green box highlighted shows occupied parking spaces, and the yellow one shows empty parking spaces. So as mentioned before, this object detection uses the ssd_mobilnet_v2_coco model as a TensorFlow model.

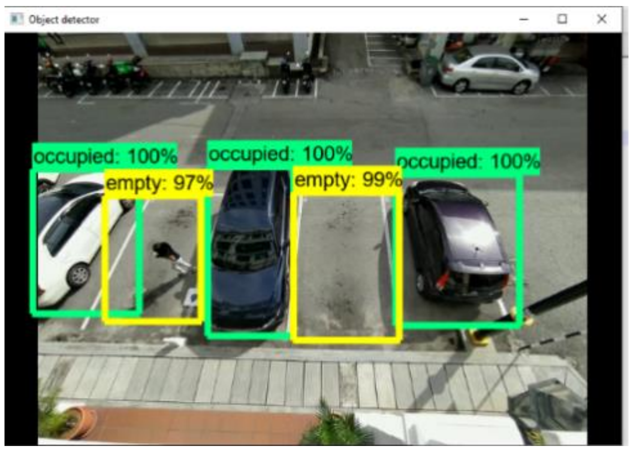


Figure 3: Video after Tensorflow object detection classifier process

i. Single Shot Detector (SSD)

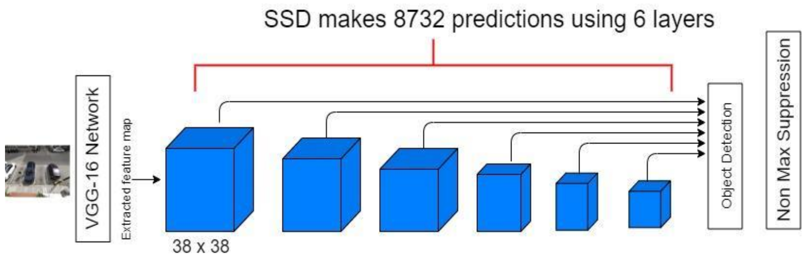


Figure 4: The architecture of Single Shot Detector (SSD)

The Single Shot Detector (SSD) layer of the architecture is shown in Fig. 4. So, this layered architecture consists of an input image. The SSD network always starts with the VGG-16 Network. The input image to be trained has passed through VGG-16 Network first. The purpose of the VGG-16 Network is to extract a feature map. The Single Shot Detector consists of 6 convolution layers that perform the object detection task classification. Single Shot Detector (SSD) makes 8732 predictions for every single object, which is for every single object the Single Shot Detector (SSD) predicted 8732 bounding boxes.

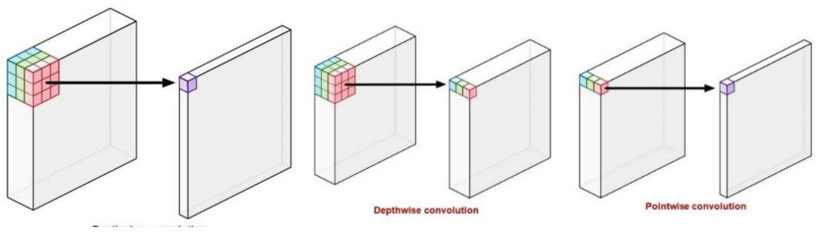
The function of the Non Max Suppression is to remove the duplicate predictions. The Single Shot Detector (SSD) checked the confidence score of each box and picked the top 200 predictions per image. Whenever giving input images to Single Shot Detector (SSD), images with ground truth boxes for each object in the image should have images. The task of 6 convolutional layers is to checkboxes of different aspect ratios at each location with different scales. The convolutional layers were scanned in multiple boxes of different sizes and aspect ratios. The amount of multiple boxes scanned by convolutional layers is 8732.

In conclusion, each object in an image with truth boxes has 8732 bounding boxes inside. The reason Single Shot Detector (SSD) using multiple boxes is for better coverage of location. So, this process obtained the box that overlapped the ground truth bounding box. The Single Shot Detector found a box with the highest overlap with the help of Intersection Over Union (IOU). So, the overlapped box appeared as output in the video feed.

ii. Tensor MobileNet

MobileNet Architecture is built on a depthwise separable filter. Depthwise Separable Convolution is used to reduce model size and complexity. Convolution is measure overlapping between two functions as one slides over the other mathematically. Mathematically it's the sum of product. The standard convolution operation is slow to perform. Fig. 5 (a) shows the standard convolution model concept.

This Standard Convolution process can speed up using Depthwise Separable Convolution. The Depthwise convolution is the channel-wise DK-DK spatial convolution. So, the MobileNet uses Depthwise Separable Convolutions, which factorise a standard convolution into depthwise convolution and follow by 1 x 1 convolution called a pointwise convolution. The difference between Standard convolution and depthwise convolution is the step taken to filtration. Standard convolution filters and combines inputs into a new form of output in one step. The Depthwise Separable Convolution separates the process into two layers: a separate layer for filtering and a separate layer for combining. Fig. 5 (b) shows the concept of the proposed system.

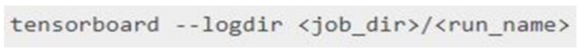


(a) (b)

Figure 5: Concept of (a) standard convolution and (b) depthwies separable convolution [12]

iii. Tensor Board Result and Discussion

During the object detection training, the training process needs to be monitored to ensure the model is functioning. Since the platform that has been implemented for this research is TensorFlow. So, the TensorFlow detection can be monitored using Tensor Board to visualise the training process. Tensor Board is a visualisation toolkit. The Tensorboard provides visualisation and tooling needed for machine learning for some experimentation such as tracking and visualising metrics such as loss and accuracy. Then it's also can be visualising the model graph, viewing histograms of weights, biases, or other tensors and much more. Fig. 6 is the code to open the Tensorboard.

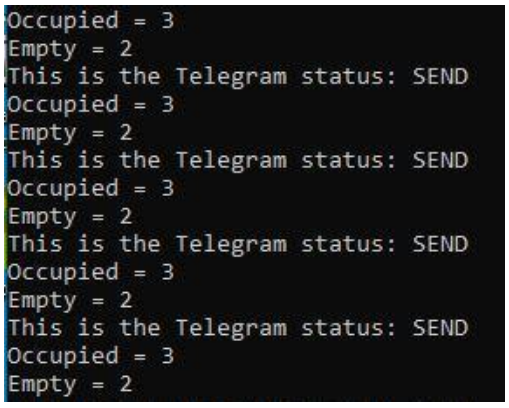


```
tensorboard --logdir <job_dir>/<run_name>
```

Figure 6: The code for Tensorboard Monitoring

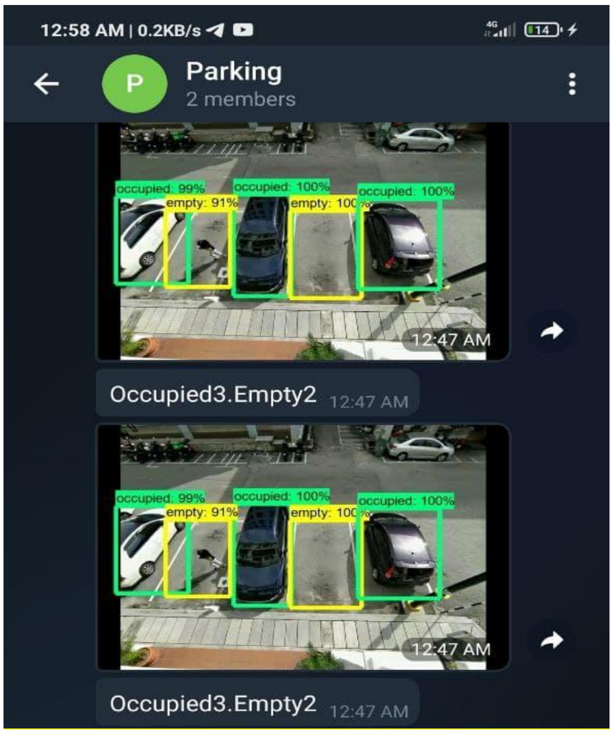
4.2 The Outcome Result for Phase 3

Fig. 7 shows the output at the command prompt of Anaconda Navigator. The total count of occupied and empty parking spaces has been keeping updated here. The main code has been added, the counter code to calculate the total number of occupied and empty parking spaces. Based on the code, the total count appears at Anaconda Prompt terminal. The main code also added Telegram push data message as mentioned previously in chapter methodology. So, the video object detection code always keeps pushing the data to Telegram as programmed to designated Bot. Fig. 8 shows the real-time monitoring status updated on Telegram Bot Group.



```
Occupied = 3
Empty = 2
This is the Telegram status: SEND
Occupied = 3
Empty = 2
This is the Telegram status: SEND
Occupied = 3
Empty = 2
This is the Telegram status: SEND
Occupied = 3
Empty = 2
This is the Telegram status: SEND
Occupied = 3
Empty = 2
```

Figure 7: The command prompt of Anaconda Navigator



Occupied3.Empty2 12:47 AM

Occupied3.Empty2 12:47 AM

Figure 8: Telegram notification system

4.3 Various Sample Video Testing on Same Trained Object Detection

Fig. 9, 10 and 11 are various video feeds used for testing the capability of the trained object detector for the parking system. As can be seen, the object detection classifier can detect occupied and empty spaces in the Fig. 9 and 10.

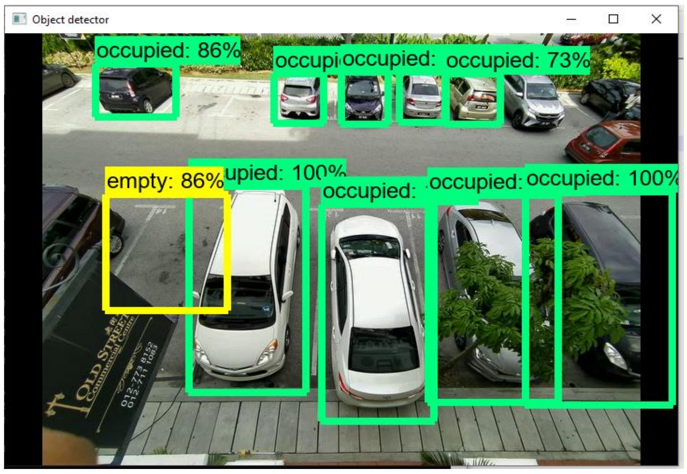


Figure 9: The first video feed sample for testing

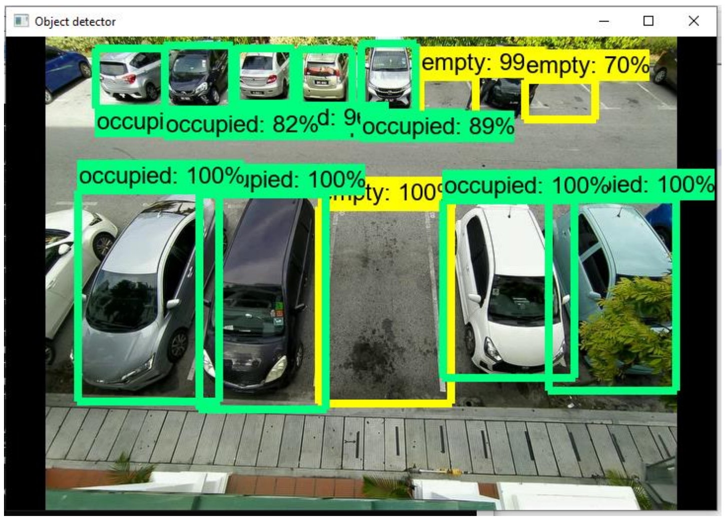


Figure 10: The second video feed sample for testing

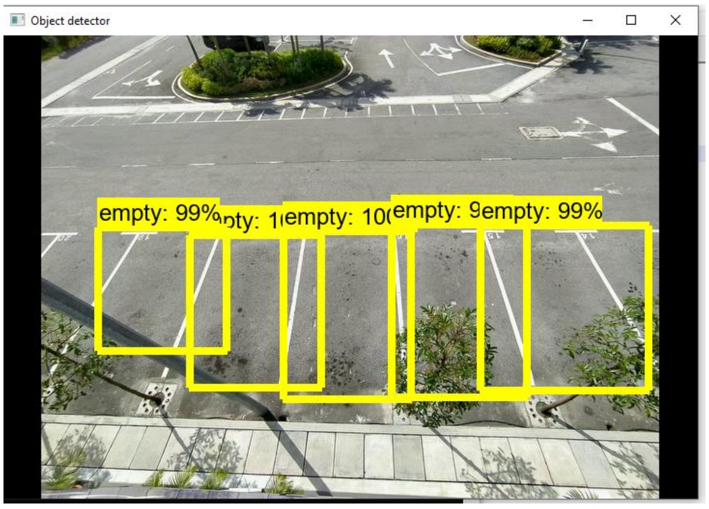


Figure 11: The third video feed sample for testing

5. Conclusion

The result obtained achieves the actual objective and some improvement can be made so that the project is satisfying. The project has been designed to make road users easier to monitor and notify the parking status of occupied and empty parking lots. The trained object detector model can use for real-time camera detection to classify occupied and empty spaces. Besides that, the detected data of parking status can push to the Telegram Bot app in the scope of the Internet of Things (IoT) system. The performance of the developed system on object detection has been successfully demonstrated. The object detection accuracy depends on the number of images used for training and the TensorFlow model. Based on the various video tested on the object detector, the trained model can successfully classify the two types of classification: occupied and empty. The advantage of this research using object detection technology is parking detection with using minimum requirement of hardware. As mentioned before, most of the existing parking system is designed for indoor areas and required hardware parts such as sensors, motor, wire configuration, and much more. So, this system can help reduce the hardware requirement. On the other hand, this project can contribute to the social contribution. This research can reduce carbon emissions, traffic congestion and time consumption. The Telegram app is the best Internet of Things (IoT) platform to implement cloud computing. Nowadays, the Telegram app is owned by almost all smartphone users. So, the user or driver of this parking system does not need to download or use a specific app on their phone.

References

[1] Wei, L., Wu, Q., Yang, M., Ding, W., Li, B., Gao, R.: Design and implementation of smart parking management system based on RFID and internet. Proc. - 2012 Int. Conf. Control Eng. Commun. Technol. ICCECT 2012, pp. 17–20, (2012). doi: 10.1109/ICCECT.2012.12.

[2] Fleyeh, H., Paidi, V., Håkansson, J., Nyberg, R.: Smart Parking Tools Suitability for Open Parking Lots: A Review (2018). doi: 10.5220/0006812006000609.

[3] Jiang, S., Jiang, H., Ma, S., Jiang, Z.: Detection of parking slots based on mask R-CNN. Appl. Sci., vol. 10, no. 12 (2020). doi: 10.3390/app10124295.

[4] Grodi, R., Rawat, D.B., Rios-Gutierrez, F.: Smart parking: Parking occupancy monitoring and visualisation system for smart cities. Conf. Proc. - IEEE SOUTHEASTCON, vol. 2016-July, pp. 1–5, (2016). doi: 10.1109/SECON.2016.7506721.

[5] Anvekar, R.G., et al.: In IoT Based System Model. IEEE Int. Conf. Technol. Innovations ICT Agric. Rural Dev., pp. 5–9 (2017).

[6] Krishna Sai, B.N., Sasikala, T.: Object Detection and Count of Objects in Image using Tensor Flow Object Detection API. Proc. 2nd Int. Conf. Smart Syst. Inven. Technol. ICSSIT 2019, no. Icssit, pp. 542–546, (2019). doi: 10.1109/ICSSIT46314.2019.8987942.

[7] Phadnis, R., Mishra, J., Bendale, S., Objects Talk - Object Detection and Pattern Tracking Using TensorFlow. Proc. Int. Conf. Inven. Commun. Comput. Technol. ICICCT 2018, pp. 1216–1219, (2018). doi: 10.1109/ICICCT.2018.8473331.

[8] Janahiraman, T.V., Subuhan, M.S.M.: Traffic light detection using tensorflow object detection framework. 2019 IEEE 9th Int. Conf. Syst. Eng. Technol. ICSET 2019 - Proceeding, no. October, pp. 108–113, (2019). doi: 10.1109/ICSEngT.2019.8906486.

[9] Cao, D., Chen, Z., Gao, L.: An improved object detection algorithm based on multi-scaled and deformable convolutional neural networks. Human-centric Comput. Inf. Sci., vol. 10, no. 1, (2020). doi: 10.1186/s13673-020-00219-9.

[10] Huang, R., Pedoeem, J., Chen, C.: YOLO-LITE: A Real-Time Object Detection Algorithm Optimised for Non-GPU Computers. Proc. - 2018 IEEE Int. Conf. Big Data, Big Data 2018, pp. 2503–2510, (2019). doi: 10.1109/BigData.2018.8621865.

[11] Wang, Y., Zheng, J.: Real-time face detection based on YOLO. 1st IEEE Int. Conf. Knowl. Innov. Invent. ICKII 2018, vol. 2, pp. 221–224 (2018). doi: 10.1109/ICKII.2018.8569109.

[12] Tensorflow - Backpropagating gradients through nested tf.map_fn - Stack Overflow. [Online]. Available: https://stackoverflow.com/questions/59094897/backpropagating-gradients-through-nested-tf-map-fn. [Accessed: 03-Jul-2021].