Car Park Occupancy Detection with Computer Vision

Batu Helvacioglu
Koc University
bhelvacioglu18@ku.edu.tr

Can Gozpinar
Koc University
cgozpinar18@ku.edu.tr

1. Abstract

Finding empty parking spaces is a problem for a lot of people in their daily lives. In urban settlements parking spaces are often limited. There are existing solutions used by shopping malls and other businesses with parking spaces, to make finding empty parking spaces easier. These solutions have space for a lot of improvement since they generally lack generalizability, robustness, cost effectiveness, and easy installment. Being inspired by the improvements on the existing solutions by relying further on the deep learning architectures, and the recent high performance of the YOLO family of models, we proposed to use YOLOv5 object detectors trained on PKLot dataset. We aim to have a model that can outperform older methods such as VGG16 [2]. YOLO family of models are lighter and more suitable for mobile applications and popular development boards. Being able to use lower end hardware for the detection task will make the overall parking space detection system cheaper. There often are security cameras that are already installed at parking lots which can be utilized for detecting parking spaces using our proposed method. Also, drones equipped with our object detection model and a mounted camera could be used to detect parking spaces where there are no cameras available to cover large complex environments.We used the PKlot dataset with data augmentations to train YOLOv5 object detectors. We achieved competetive results with the existing solutions on the PKlot dataset. PKlot is a relatively simple dataset and our models overfitted as expected. However we believe our data augmentations improved the generalizability of our models compared to using non-augmented PKlot for training based on our visual observations of our model's predictions on drone footage.

2. Introduction

In our project we tackled the problem of detecting empty and occupied car parking spaces given images and videos. Finding empty parking spaces is a problem for a lot of people in their daily lives. The steep growth of the population has resulted in a decrease in the empty parking spaces. This results in traffic congestion, excess consumption of fuel and the frustration of the drivers. The currently available solutions to car parking space occupancy detection lacks in many ways since they are not cost effective generalizable solutions which could cover large complex areas that change relatively fast in time. We are offering a deep learning based computer vision model which could directly yield bounding box coordinates and class labels given images for the empty and occupied car parking spaces that appear in the frame. We are motivated by the papers [2] [1] who performed car parking space occupancy detection using older deep learning architectures and the high performance of the recent YOLO family of models performance. Our proposed method uses YOLOv5 object detectors trained using PKLot dataset [4] to perform car parking space occupancy detection. Our project primarily targets the parking space operators and the shopping malls with huge parking space capacities. We have evaluated our trained models on both a training set gathered from the PKLot dataset [4], and on videos that were collected by us using DJI MAVIC MINI drone in our home town Atasehir, Istanbul. Even though the performance on our drone's footage did not yield as accurate results as it did on the test split from the PKLot dataset, we had promising results. Our experience has lead us to conclude that upon gathering a more diverse training dataset than PKLot [4], one could end on a feasible alternative to the existing car occupancy detection solutions.

3. Related Work

The existing work surrounding our area of interest can be separated into hardware intense solutions and computer vision based solutions. To clearly observe the value that deep learning could bring to this field, let us go through some of the hardware intense existing solutions.

3.1. Existing Hardware Intense Solutions

These existing solutions often include sensors for each parking space or a counter to show how many cars in total are in the parking lot at the moment and how many empty spaces are left. Counter solution is cheaper but it lacks precision and it does not indicate where the empty spaces are located in the given vicinity. On the other hand, sensors located at each parking space can give more precise information to people, but they come at a significant cost of integrating many sensors and high maintenance costs. Sensor based solutions require area specific installation of sensors and their corresponding infrastructure. Maintaining these sensors is another issue that rises commonly due to sensory malfunctions. Additionally, these solutions require modifications to the already existing infrastructure upon any change in the car parking spacing.

3.2. Computer Vision Based Solutions

On the other hand, computer vision based solutions are easier to deploy, more robust and more customizable. There has been many attempts at implementing computer vision based object detectors for our task and following are the some that stood out to us.

3.2.1 Classification of Manually Marked Parking Spaces

One primitive approach is to draw bounding boxes for each parking space in a stationary camera's field of view and then crop these boxes to feed them into a simple car, no-car image classification model [3]. The classification outputs yields the status of the corresponding parking spaces. This approach works quite well but it requires the user to manually set up the bounding boxes of the parking spaces for every camera. Even though this would be a one time only set up, it prevents the system from being used in non-stationary cameras such as the ones mounted on drones, and any disturbance to the camera's initial set up position would result in a system dysfunction.

3.2.2 Inference from Separate Detection of Parking Spaces and Cars

In order to tackle these issues, there have been some attempts at training models to directly recognize the occupied and empty parking slots [5] [2] [1]. These first detect the parking spaces and the cars that are apparent in the given frame [6]. Then, these results are used to determine the occupied and empty car parking spaces [6]. This is not very efficient since it applies an approach named intersection of union on the outputs of the object detector. Bounding boxes of every car detection is compared against every car parking space detection and based on the intersection of their bounding box area's, they are classified as empty or occupied parking spaces [6]. Also, the papers that implemented this approach did not share their test results on data, that is different from their training data. They used different images from the same cameras in their testing and training datasets. Since the location of the parking spaces do not change in these images, it is not clear to us if the object detector just memorized the parking spaces or if it actually learned something generalizable.

3.2.3 Initial Detection of Parking Spaces and Subsequent Classification

There was another approach that depended further on deep learning architectures and eliminated the need for this rule based decision mechanism named intersection of union. This method first runs the image through an object detector that detects parking spaces regardless of whether they are occupied by a car or not [5]. Later, according to the yielded bounding boxes, patches that correspond to these parking space locations are cropped and they are individually fed into a classifier which returned a label of whether they were empty or not [5]. This method requires more resources and longer processing times. This makes this model very inefficient for deployment in popular development boards and mobile devices. Moreover, usually it could not identify occupied car parking spaces whenever a parked car had obstructed the parking space.

3.2.4 Direct Detection of Occupied and Vacant Spaces

This brings us to our last existing approach . To address these drawbacks, another approach is to directly train object detectors to detect and classify car parking spaces. Similar to this approach, YOLOv3 which is another deep learning based computer vision architecture was trained on the PKLot dataset[4] , which we also used, and it achieved promising results [1]. We are inspired by the performance of VGG16 [2] on another popular public parking occupancy detection dataset.

4. Data

For the training of our object detectors we decided to use PKLot dataset [4]. It is comprised of around twelve thousand images which embodies approximately seven hundred thousand parking spaces. These images are captured from two different parking lots with each having two different cameras installed. We believe that having a total of four different camera angles really limited our models generalizing capabilities which we tried to counteract by leveraging the benefits of data augmentation. On the other hand, images are taken under various weather conditions such as sunny, cloudy and rainy days, which could help with the generalizibility of our model. Every image in the dataset includes the bounding box coordinates and the corresponding class labels as occupied or vacant for every car parking space available in a given image. We experimented with several prepocessing steps in our project. These include resizing the images to 640x640 pixels to reduce dataset size, using 2x2 tiles to ease the detection of small objects and grayscale to reduce training time as well as eliminating color to be able to generalize better in parking lots that have different colored floors and cars with exotic colors. The combinations of preprocessing and data augmentations we chose will be explained further in the methods and experiments sections.

5. Methods

5.1. Computer Vision Models

One of the main aim of our project was to get a computer vision model that can generalize well to new environments that it had not seen before. Inline with this objective we eliminated the manually marking parking spaces approach. Even though this method would have been the most accurate approach, it was not possible to integrate it with a drone which was another key point of our project. This approach required stationary cameras to be able to mark the parking spaces. Another approach we had to eliminate was the separate detection of parking spaces and cars due to the computational requirements of this method. Using intersection over union on parking space and car bounding boxes introduces added computations on top of the two separate machine learning models needed for the detection of parking spaces and cars. Since we wanted to use a drone for the parking lot occupancy detection task, the computer vision model needed to be computationally lightweight and have fast inference times. Drones often cannot accommodate powerful enough hardware to run heavy models. We could have streamed the video feed to a computer and run the inference on the computer but we did not want to introduce extra complexity and costs to our solution. Therefore the method we chose must be able to run on relatively lightweight hardware such as Nvidia's Jetson Nano developer kit. A computationally light model will also decrease the battery life of the drone less. Therefore we decided to implement the directly detecting empty and full parking spaces approach. Single shot detectors such as the YOLO family are suitable for this task as they perform localization and classification in a single pass. This lines up with our computationally lightweight and generalizable requirements. We wanted to improve up on the results of the study that used YOLOv3 for this task. Since by modern day standard YOLOv3 can be considered outdated as it was released in 2018, we decided to approach the problem with YOLOv5 which was much newer being released in 2020. YOLOv5's ease of integration for inference was also an important factor on our decision.

5.2. Data Augmentation Methods

Other methods that were essential for our project were data augmentations. The existing parking lot datasets PKlot and CNRParkingExt were not varied enough for our use case. These datasets only offer images from a limited number of cameras. The studies that used these datasets often report high performance on the datasets. However we believe these results are misleading since the training and test images are from the same set of cameras. This causes the computer vision models to overfit and memorize the locations of the parking spaces instead of learning to detect parking spaces. There only remains classifying if the parking space is empty or not which can also be done with a simpler image classifier instead of an object detector. Furthermore we want to integrate our computer vision model to a drone which would have its camera in totally different angles compared to stationary cameras. To solve these problem we decided on augmenting the PKlot dataset. We experimented with image level and bounding box level data augmentations. For the bounding box level augmentations, we used 90 degree rotation, crop, horizontal flip, +-15 degrees of rotation and +-15 degrees of sheer. Our main focus with these augmentations were to simulate different camera angles that our drone might have. In addition we believed cropping could help the model generalize by changing the location of the bounding boxes and therefore preventing the model from memorizing. We also experimented with image level augmentations. Specifically +-15 degrees of rotation to accommodate for camera roll and +-15 degrees of sheer to accommodate for camera perspective as well as subject pitch and yaw. Augmenting training data also increased our sample size significantly as we were able to generate around 27 thousand new images from the original PKlot dataset.

6. Experiments

We used the mean average precision at 0.5 percent overlap and mean average precision at 0.95 percent overlap metrics to compare the results of our experiments. All of our experiments except 1 used YOLOv5 small model. We experimented with YOLOv5 medium once. Big part of our experimentation was on the data augmentation and preprocessing the dataset side of things. We decided to modify the dataset instead of the model hyperparameters since regardless of the hyperparameters, our models overfitted the original PKlot dataset in our preliminary tests. YOLOv5 Medium model got 0.72298 mAP@0.95 and 0.97096 mAP@0.5 in 4 epochs and the small model got 0.8314 mAP@0.95 and 0.98782 mAP@0.5 in 12 epochs. Our first experiment consisted of adding resizing and 2x2 tile steps to preprocessing and all the bounding box level augmentations that we have mentioned. We trained a YOLOv5 small model that was pretrained on COCO dataset for 200 epochs with a batch size of 32. The results we got were almost an order of magnitude worse than our preliminary testing of 11 epochs. We suspected that the dataset became too complex for YOLOv5 small to learn. With this assumption we switched to YOLOv5 medium with batch size 16 for 200 epochs again pretrained on COCO dataset like all the models in our experiments. YOLOv5 medium managed to get a significantly better performance compared to YOLOv5 small. MAP@0.5 was around 20 percent higher and MAP@0.95 was 13 percent higher. However YOLOv5 medium took significantly longer to train, so we decided to experiment with preprocessing and augmentations to increase the performance of YOLOv5 small model. After inspecting the training images we realized that resizing the images to 640x640 pixels made the images pixelated and some of the cars lost details since they occupy a small area in the image due to being far away from the camera. Keeping in mind that YOLOv5 already resizes images to 640x640, we decided that the resizing preprocessing step was not needed. We trained another YOLOv5 small model without resizing and got results close to the YOLOv5 medium model. YOLOv5 small without resizing got only 5 percent lower than the medium model. We decided to continue with YOLOv5 small without resizing instead of medium in order to be able to do more experiments. We later dropped the crop and 90 degree rotation bounding box level augmentations since our primary focus was to generalize for the drone's camera angles. This resulted in a 15 percent increase in MAP@0.5 and 10 percent increase in MAP@0.95. Dropping the horizontal flip bounding box level augmentation further improved results by around 5 percent both in MAP@0.5 and MAP@0.95.We also decided at this step of our experimentation that 100 epochs was enough by looking at Figure 1. It can be seen that validation loss did not improve much after the 100 epoch.

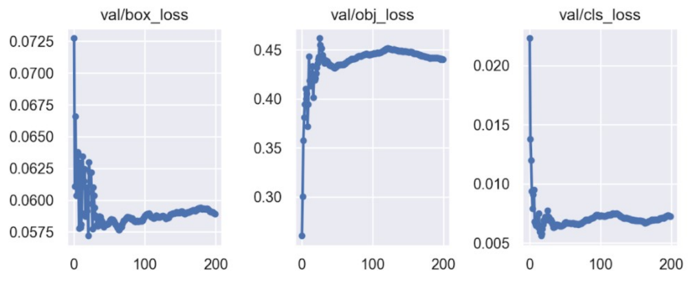
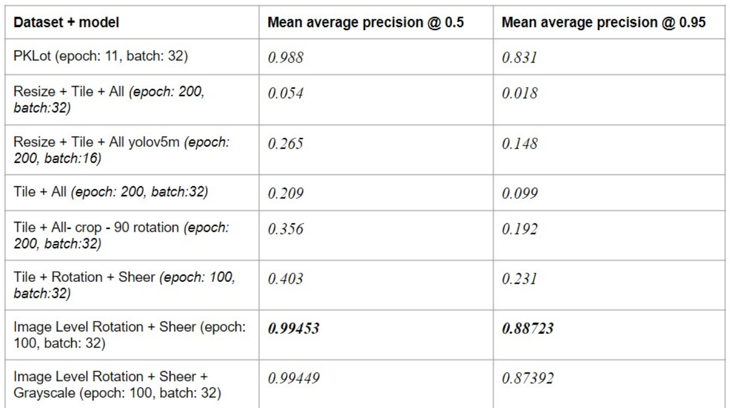


Figure 1. Validation losses of YOLOv5 Small Epoch: 200 Batch: 32 on Tile + ALL - Crop - 90 rotation

After analyzing the training images further we decided that 2x2 tile and bounding box level augmentations were making the dataset too complex. We also realised that 2x2 tile was not needed to improve performance on small objects since YOLOv5 already did mosaic augmentation automatically to achieve the same task. Tile preprocessing could still have improved training time but we belive it complicated the dataset too much when combined with YOLOv5 native mosaic augmentation. We switched to image level augmentations of +-15 degrees of rotation and sheer to simulate camera angles. This simplified the dataset and improved performance drastically. We managed to get 99 percent MAP@0.5 and 88.7 percent MAP@0.95.



Dataset + model	Mean average precision @ 0.5	Mean average precision @ 0.95
PKLot (epoch: 11, batch: 32)	*0.988*	*0.831*
Resize + Tile + All *(epoch: 200, batch:32)*	*0.054*	*0.018*
Resize + Tile + All yolov5m *(epoch: 200, batch:16)*	*0.265*	*0.148*
Tile + All *(epoch: 200, batch:32)*	*0.209*	*0.099*
Tile + All- crop - 90 rotation *(epoch: 200, batch:32)*	*0.356*	*0.192*
Tile + Rotation + Sheer *(epoch: 100, batch:32)*	*0.403*	*0.231*
Image Level Rotation + Sheer (epoch: 100, batch: 32)	***0.99453***	***0.88723***
Image Level Rotation + Sheer + Grayscale (epoch: 100, batch: 32)	*0.99449*	*0.87392*

Figure 2. Results of our experiments

We know that our model overfitted to the dataset however we believe it is better at generalizing than the model trained on the non-augmented dataset. We acknowledge that the model trained on the non-augmented PKlot dataset was trained only for 11 epochs and should have been trained for 100 epochs at least for a better comparison. This was due to Google Colab timing out during our preliminary testing with the non-augmented dataset. However even in 11 epochs it got similar results to our augmented model that trained for 100 epochs. If we look at Figure 3 and 4, screenshots from the non-augmented model and image level augmented model respectively, we can see that image level augmented model performs better in unknown datasets.

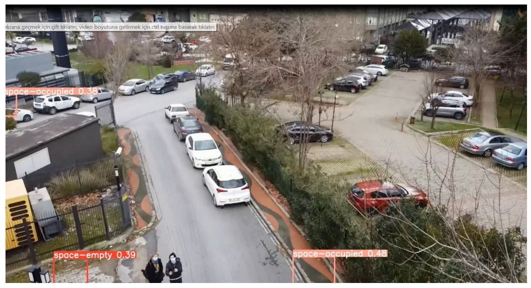

ekrana geçmek için çift tıklatın, video boyutuna getirmek için ctrl tuşuna basarak tıklatın
space—occupied 0.38
space—empty 0.39
space—occupied 0.48

Figure 3. Our custom object detector's output on a never seen before video frame gathered by using our DJI MAVIC MINI in Atasehir, Istanbul[red boxes: vacant, pink boxes: occupied]

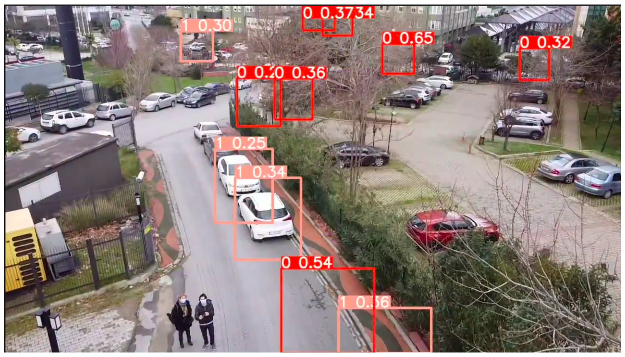


Figure 4. Our custom object detector's output on a never seen before video frame gathered by using our DJI MAVIC MINI in Atasehir, Istanbul[red boxes: vacant, pink boxes: occupied]

We have noticed that our computer vision model struggled with detecting cars that are directly under the drone as it can be seen in Figure 5. We believe this is due to PKlot dataset not having any cameras that see cars from directly above. All the data we have are at an angle.

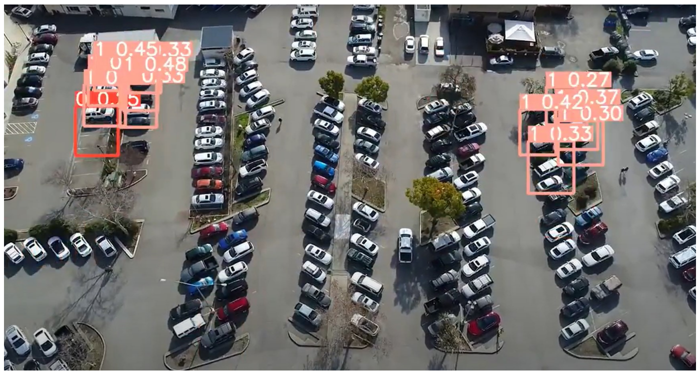


Figure 5. Screenshot from Image Level Augmented YOLOv5 Small Epoch:100 Batch:32

Another interesting finding was there were some unlabelled parking spaces in a validation image and our model managed to detect the unlabelled parking space as it can be seen from Figure 6 and 7.

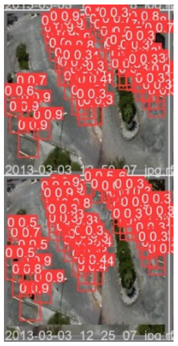


Figure 6. Prediction on Validation set of Image Level Augmented YOLOv5 Small Epoch:100 Batch:32

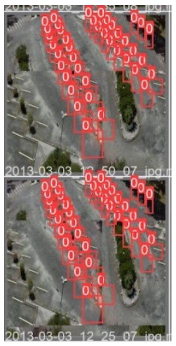


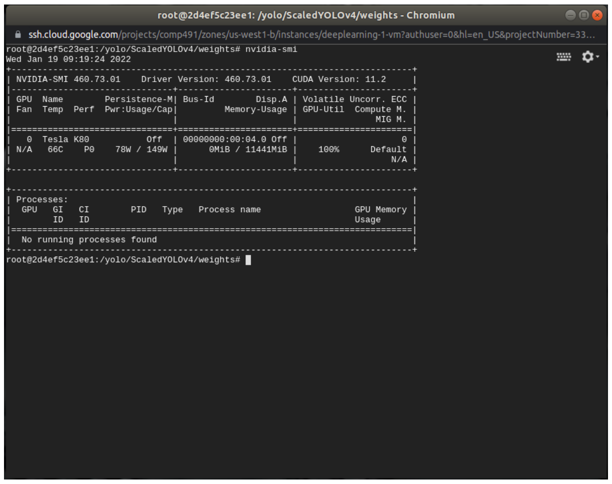
Figure 7. Image from validation set with missing parking space labels

7. Conclusion

We managed to generalize better with our augmented datasets compared to using non-augmented PKlot. However our object detectors still does not generalize well enough to be used in a drone due to the limited camera angles in the data. As we have shown cars that are right below the drone are not detected by our computer vision model. This problem can solved with collecting data with a drone. However this introduces a new problem of labelling data. It is a difficult and time consuming task to label all the frames from a drone video footage that is moving. Selecting certain frames from the video footage might elevate this problem. Using the CNRPark-Ext dataset alongside PKlot could introduce a variety of new cameras to the dataset. However we are doubtful that this would make the dataset varied enough as camera angles from both of the datasets are quite similar and do not include angles a drone would encounter. Also the CNRPark-Ext dataset has a different labeling format which give cropped images of parking spaces. A script would be needed to convert CNRPark-Ext labels to YOLO txt label format. Looking at the model perspective, our models can be further improved by using YOLOv5 medium or YOLOv5 large models. We have shown that dataset with bounding box level augmentations is too complex for YOLOv5 small but works better with YOLOv5 medium. With enough GPU resources bounding box level augmentation may generalize better than image level augmentations on larger models. Other object detectors can also be tested such as Scaled-YOLOv4. We planned on experimenting with Scaled-YOLOv4 but A simpler solution to the car parking occupancy detection with drones problem could be by painting a certain color circles in parking spaces which would only be visible when the parking space is empty. It would be a lot simpler to detect for example a green circle compared to empty or full parking spaces. After the detection the drone can navigate to the parking space and relay GPS info or some sort of location triangulation with Wifi access points if there is no GPS signal in the area.

8. Supplementary Material

- https://hub.docker.com/r/cgozpinar18/scaled_yolov4
- https://hub.docker.com/r/cgozpinar18/drone_object_detector
- https://public.roboflow.com/object-detection/pklot
- https://github.com/ultralytics/yolov5



root@2d4ef5c23ee1: /yolo/ScaledYOLOv4/weights - Chromium

ssh.cloud.google.com/projects/comp491/zones/us-west1-b/instances/deeplearning-1-vm?authuser=0&hl=en_US&projectNumber=33...

```
root@2d4ef5c23ee1:/yolo/ScaledYOLOv4/weights# nvidia-smi
Wed Jan 19 09:19:24 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.73.01    Driver Version: 460.73.01    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   66C    P0    78W / 149W |      0MiB / 11441MiB |    100%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
root@2d4ef5c23ee1:/yolo/ScaledYOLOv4/weights#
```

Figure 8. Google Cloud Deep Learning VM's GPU used during training our custom models

References

[1] "Asian Institute of Technology." [Online]. Available: http://ise.ait.ac.th/wp-content/uploads/sites/57/2020/12/Car-Parking-Occupancy-Detection-using-YOLOv3.pdf. [Accessed: 22-Nov-2021].

[2] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo, "Deep learning for decentralized parking lot occupancy detection," Expert Systems with Applications, vol. 72, pp. 327–334, 2017.

[3] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, "Car parking occupancy detection using Smart Camera Networks and deep learning," 2016 IEEE Symposium on Computers and Communication (ISCC), 2016.

[4] Almeida, P., Oliveira, L. S., Silva Jr, E., Britto Jr, A., Koerich, A., PKLot – A robust dataset for parking lot classification, Expert Systems with Applications, 42(11):4937-4949, 2015.

[5] V. Visualbuffer, "Visualbuffer/Parkingslot: Automated parking occupancy detection," GitHub, 11-Aug-2020. [Online]. Available: https://github.com/visualbuffer/parkingslot. [Accessed: 30-Dec-2021].

[6] Chen, L.-C., Sheu, R.-K., Peng, W.-Y., Wu, J.-H., &amp; Tseng, C.-H. (2020, February 6). Video-based parking occupancy detection for smart control system. MDPI. Retrieved January 20, 2022, from https://www.mdpi.com/2076-3417/10/3/1079/htm