American University of Armenia

# REAL-TIME PARKING OCCUPANCY DETECTION

*Author:*

Laura Barseghyan

*Supervisor:*

Elen Vardanyan

*A project submitted in fulfillment of the requirements for the degree of BS in Data Science*

*in the*

Zaven & Sonia Akian College of Science and Engineering

May 5, 2023

## Abstract

Nowadays, finding vacant parking spaces in urban areas has become a problem. Drivers spend a lot of time finding vacant parking spaces, which is stressful and results in traffic congestion and increased fuel consumption. To solve this problem, we developed a deep learning-based system that can accurately detect and classify parking spaces as occupied or unoccupied in real-time, using live video feeds from cameras installed in the parking lot. The suggested system consists of two components: object detection using YOLO [1] and parking space occupancy detection using IoU(intersection over the union). To train and test our object detection model, we also created and labeled our dataset, which consists of images taken from the recordings of the cameras installed in a parking lot. Testing results showed that our system works with high accuracy and can be applied in real-life situations.

## 1. Introduction

The need for efficient and convenient parking systems has become increasingly important in urban areas with the growth of population and the increase in vehicle ownership. Finding a vacant parking space in a crowded parking lot can be frustrating and time-consuming for drivers, resulting in traffic congestion and increased fuel consumption.

The traditional parking management systems that rely on manual intervention often lead to both inaccuracies and inefficiencies, further adding to the problem. Addressing these challenges requires innovative approaches, and a viable solution can be provided by automated parking occupancy detection systems. To build a robust and efficient parking occupancy detection model, the latest deep learning, computer vision, and real-time processing techniques are used.

We developed a deep learning-based model that can accurately detect and classify parking spaces as occupied or unoccupied in real-time, using live video feeds from cameras installed in the parking lot. Our system consists of two main parts: object detection and parking space occupancy detection. The first part of the system takes input from the video feeds and detects the cars, and the second part decides whether the parking space is available by calculating the intersection over the union between the parking space and each detected car.

To develop the system, we used a dataset of videos from cameras installed in a gas station in Armenia. Because the model is trained on this specific dataset, it works very accurately for this gas station’s parking lot and for other parking datasets with reasonable accuracy. After slight modifications and fine-tuning, it will work with higher accuracy for other parking lots as well.

After creating a user interface, the developed system can be used to provide drivers with real-time parking occupancy status, enabling them to quickly find vacant spots and reduce traffic congestion. Furthermore, the system can assist parking management authorities in monitoring parking occupancy in real time and optimizing the use of parking spaces.

The remainder of this paper is organized as follows. **Section 2** provides an overview of related works in parking occupancy detection. **Section 3** provides information about the tools and technologies used to develop the system. **Section 4** describes the data used in the project and explains the preprocessing steps for cleaning and preparing the data for the subsequent analysis. **Section 5** describes the methods and techniques used and gives a detailed explanation of how the system works. **Section 6** presents the experimental results and discusses the performance of the proposed approach. Finally, **Section 7** concludes the paper and outlines future work.

## 2. Literature Review

The problem of finding an available parking space is not new, it has been around for many years. Over time, various solutions have been suggested for this problem and already in recent years, the deep learning-based approaches have shown significant promise. This literature review summarizes some of the recent developments in parking occupancy detection using different deep-learning methods.

Acharya et al. [2] developed a real-time image-based parking occupancy detection system for outdoor parking spaces. The technique combines a deep Convolutional Neural Network (CNN) and a binary Support Vector Machine (SVM) classifier. The classifier was trained and put to the test using features that the deep CNN learned from the publicly available PKLot dataset with various lighting and weather conditions. They achieved a high detection accuracy rate of 99.7% for the public dataset and 96.7% for their dataset.

Xiangwu Ding and Ruidi Yang [3] proposed an improved CNN model called You Only Look Once (YOLO) for vehicle and parking space detection. They enhanced YOLOv3 [4] by incorporating a residual structure to extract deep features of vehicle parking spots. The outcomes of the experiments reveal that their approach can enhance the detection precision of both vehicles and parking spaces while simultaneously decreasing the detection rate.

YOLOv3 was also used by Chen et al. [5]. They proposed a technique that can recognize the availability of parking spaces on streets and regulate the streetlights to aid in this detection process. Researchers utilized YOLOv3 object detection, along with MobileNetv2 [6] and object overlapping identification, to accurately identify the occupancy status through a voting mechanism. To verify their solution, they tested it using the CNRPark + EXT [7] dataset, a simulated model, and real-life scenarios captured by a camera.

Naufal et al. [8] proposed another smart parking system to determine the availability of vacant parking spaces. The system consists of two stages; the first involves marking the parking positions on the input image of an entire parking lot using Mask R-CNN [9] and Exposure Fusion [10]. The second stage examines each parking position using mAlexNet [11] to determine if it's available or not. The system achieved an accuracy of 85.80% for marking parking positions and 73.73% for determining parking space availability on video data.

Scekic et al. [12] proposed an image-based parking occupancy detection system using Faster R-CNN [13] and Detectron2 [14] software library. They used the publicly available dataset PKLot [15] and achieved a precision of around 93%.

In the abovementioned papers, developers used different techniques for solving parking occupancy detection. Most of them used YOLOv3 and different types of R-CNN models and got outstanding results in terms of accuracy.

## 3. Tools and Technologies

During the implementation of the parking occupancy detection system, we used several tools and technologies, which are described below.
The programming language that we used for the project is Python. To write and test the code, we used PyCharm, Jupyter Notebook, and Google Colab [16]. Google Colab was primarily used for training the object detection model as it is a cloud-based environment that also provides free access to GPUs. From the Python libraries, we mainly used NumPy [17] for array processing, OpenCV [18] for computer vision related tasks, and Ultralytics [19] for training and testing the object detection model. To label our training data, we used Roboflow [20], which is a platform for building and deploying computer vision models, providing tools for labeling, augmenting, and preprocessing images.

## 4. Data

To create a robust parking occupancy detection system, we first needed a parking lot dataset that would include photos of the parking lot taken over several days with different weather conditions. For this purpose, we requested video recordings of several days from a gas station in Armenia. We were provided with video recordings from cameras installed in the parking lot of the gas station. We separated videos of three days with different weather conditions: sunny, rainy, and foggy (Figure 1). This was done to have a variety of weather conditions in the dataset, as this will ensure the accuracy of the train model for different weather conditions.
We took 23 videos of 30 seconds from each day's recordings and made sure to have these videos from different times of the day, from morning to evening. Then we took one frame per second from the videos, and as a result, we got 2102 images in total. Then this was divided into the train, validation, and test dataset. The training dataset has 1668 images overall, and the validation and test datasets have 217 images each.

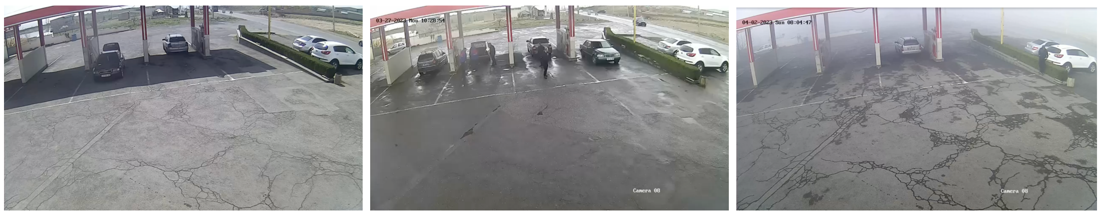

(a) (b) (c)

Figure 1. Photos of the parking lot over several days with different weather conditions: (a) sunny, (b) rainy, (c) foggy.

For training an object detection model, we labeled our data with the help of the Roboflow tool. We drew tight bounding boxes around all the vehicles that could park in the parking lot, this includes cars, trucks, and buses. Roboflow has options of different formats for exporting the dataset, and one of them was explicitly for YOLOv8 [21], which is the object detection model that we trained using this dataset. Hence, we used the YOLOv8 format to download the dataset. The final dataset consists of three folders: train, test, valid, as well as the `data.yaml` file. Train, test, and valid folders contain two folders each, one for images and one for keeping the information about the bounding boxes. YOLOv8 keeps a text (`.txt`) file for each image, where each line has information about one bounding box. In the text file, each bounding box is represented in the following format:

```
<class> <x> <y> <w> <h>,
```

where `<class>` is the object's class id, `<x>` and `<y>` are the coordinates of the bounding box's center, and `<w>` and `<h>` are the width and height of the bounding box, respectively. The final dataset also includes the `data.yaml` file, which contains the path of the train, test, valid directories and the class names.

## 5. Method

Upon reviewing the literature, we identified several methods for implementing automated parking space detection. After evaluating these methods, we selected a highly accurate approach to serve as the foundation for our system. Our approach consists of two primary components: object detection and parking occupancy detection. Each of these components will be discussed separately in the subsequent sections.

### 5.1 Object Detection

Object detection is a technique in computer vision that allows us to identify and locate objects within a video and an image. To do this, object detection methods draw a bounding box around each object. This allows the system to recognize the object and its spatial location within the image. There are various algorithms for object detection, each with its strengths and weaknesses. The most popular ones in parking occupancy detection are YOLO and R-CNN with its varieties such as Faster R-CNN or Mask R-CNN. During the development of our system, we first tried Mask R-CNN and then YOLO. After comparing them and testing for some videos from our dataset, we noticed that YOLO is a far better choice for our system.

The most important reason for choosing YOLO is its faster inference time than Mask R-CNN. YOLO uses a single neural network to perform both object detection and classification, whereas Mask R-CNN uses a two-stage process that involves region proposal and classification. With fewer computations required, this makes YOLO more efficient. This hypothesis has been confirmed by our empirical results on our test dataset. As a simple experiment, we ran inference on the Mask R-CNN model, which processed one frame in 4 seconds on average, whereas the YOLO (version 8) model detected one frame in 0.2 seconds on average. This shows that YOLO is around 20 times faster than Mask R-CNN, which is particularly important in our case, where the system must identify parking spot status in real time.

Additionally, the newer versions of YOLO can perform just as well and sometimes even better than Mask R-CNN models in terms of accuracy. It is also important to note that compared to the R-CNN, Mask-RCNN, or Faster R-CNN models, the YOLO family of methods gets updated frequently, with more efficient and better-performing models being published every few months.

Overall, YOLO's speed, efficiency, and accuracy make it a strong choice for real-time parking occupancy detection.

### 5.1.1 YOLO

YOLO (You Only Look Once) is a widely used object detection algorithm. It was introduced in 2016 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi and has since become one of the most popular used methods in computer vision. YOLO achieved state-of-the-art results by taking a fundamentally different approach to object detection, outperforming other real-time object detection algorithms by a significant margin.

The YOLO algorithm employs a straightforward deep CNN (Convolutional Neural Network) to identify objects within an image. The structure of the convolutional neural network that serves as the backbone of YOLO is illustrated in Figure 2.

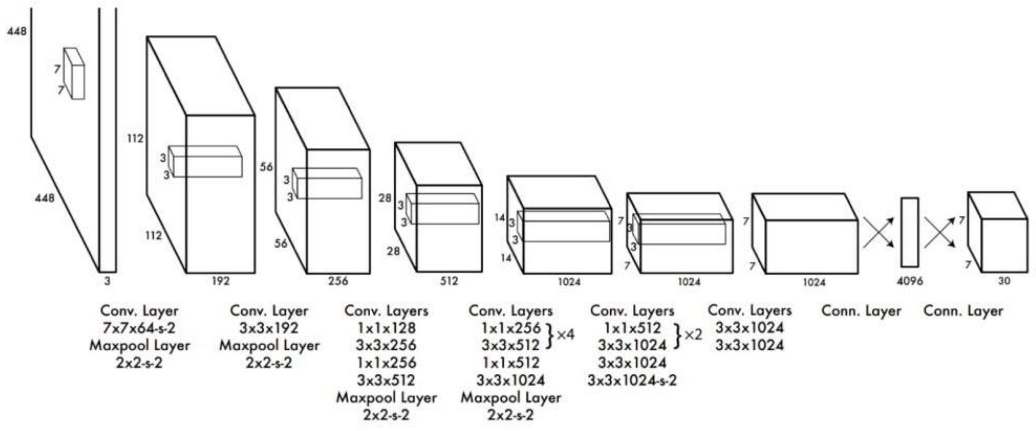


Figure 2. The architecture of the convolutional neural network that serves as the backbone of YOLO

The first 20 convolution layers of the YOLO model are trained using ImageNet by inserting a temporary average pooling and fully connected layer. Afterward, the model is converted to perform detection since research demonstrated that a pre-trained network's performance can be improved by adding convolution and connected layers. YOLO's final fully connected layer predicts both bounding box coordinates and class probabilities.

YOLO starts object detection by splitting an input image into a grid of N × N cells. If an object's center falls within a cell, that cell is in charge of identifying the object. Every grid cell predicts B bounding boxes with their confidence scores. These confidence scores indicate how certain the model is that the bounding box includes an object and how precise it believes the predicted box is.

YOLO predicts several bounding boxes for each cell. During training, only one bounding box predictor is assigned to each object. YOLO selects the predictor with the highest current IoU (Intersection over Union) with the ground truth as the "responsible" predictor. Each predictor becomes better at predicting specific object sizes, aspect ratios, or classes, improving the recall score.

To enhance object detection's accuracy and efficiency, YOLO models use non-maximum suppression (NMS). For a single object in an image, multiple bounding boxes may be produced, which might overlap or be located at different positions. NMS is applied to eliminate duplicated or inaccurate bounding boxes and generate just one bounding box for each object in the image. Object detection steps for YOLO models are illustrated in Figure 3.

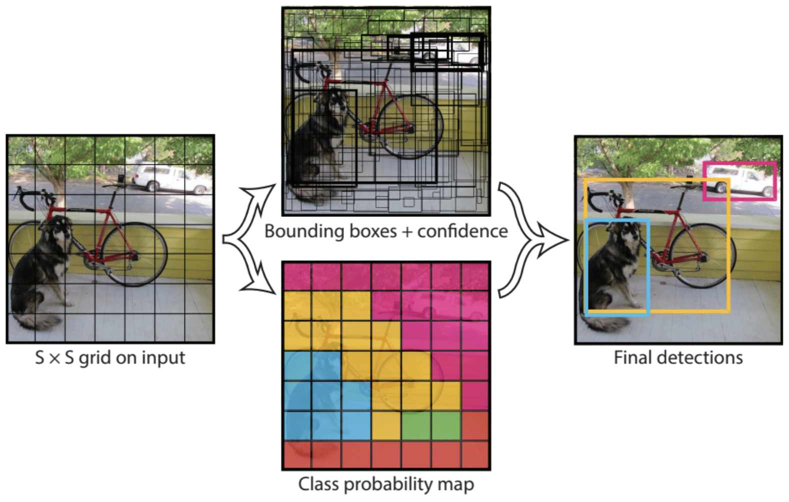


Figure 3. Object detection steps of YOLO models

The latest version of the YOLO model, YOLOv8, released in January 2023, is a state-of-the-art model for classification, detection, and segmentation tasks created by the Ultralytics team. It is licensed under the GNU General Public License, allowing users to freely distribute, modify, and share the software.

YOLOv8 has several important features that distinguish it from earlier versions.
The critical features of YOLOv8 include an updated backbone network that employs EfficientNet to enhance the model's high-level feature-capturing ability. Additionally, a new feature fusion module integrates features from various scales to improve object detection. Furthermore, YOLOv8 has implemented enhanced data augmentation techniques. All of these features result in improved accuracy and speed in comparison to earlier versions of YOLO (Figure 4).

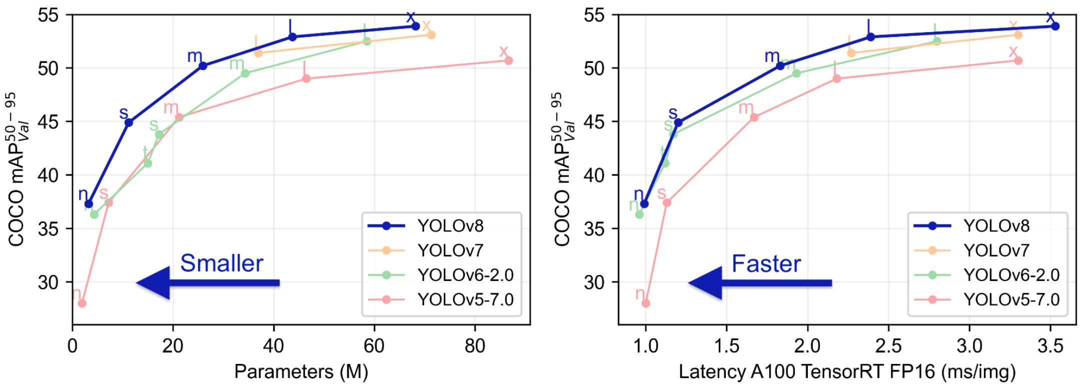


Figure 4. Performance comparison of YOLO models

In YOLOv8, there are five models of different sizes: nano, small, medium, large, and extra large. The size of the model is directly proportional to the mean average precision (mAP) and inversely proportional to the time it takes for inference. Larger models require more time for accurate object detection with higher mAP, while smaller models have faster inference times but comparably lower mAP.

### 5.1.2 Training YOLOv8

For parking occupancy detection, we needed a model that could make real-time and accurate predictions. Among the YOLO versions, the latest version is the fastest and the most accurate one, so we used the 8th and latest version.

As we have already mentioned, YOLOv8 has models of different sizes. For our experiments and future use, we chose the smallest YOLOv8n (nano) model mainly because of its speed, since it is naturally much faster than the larger models. In terms of accuracy, on the other hand, the baseline nano model is said to be inferior to the larger models. However, after training the small model on our dataset, we got satisfactory results.

To achieve good performance, we used a model that has already been pre-trained on the COCO [22] dataset. The COCO dataset is a large-scale labeled image dataset with around 330,000 images and 80 categories of objects. Using the pre-trained YOLOv8 model, we continued fine-tuning it on our dataset. The advantage of using the pre-trained model is that it has acquired the ability to recognize particular patterns and features in images, and it applies that understanding during the subsequent training procedure.

For training, we kept the majority of the default parameters set by the Ultralytics package, as they were appropriate for our case as well. One thing that we changed in the default training mode is that we trained the model as a single-class model. Initially, we had three classes in our dataset: bus, car, and truck, but by training the model as a single-class model, we now have only one class that encompasses all the three types of vehicles mentioned above.

## 5.2 Parking Space Occupancy Detection

In this section, we will explain how our parking occupancy detection system works. The trained model is an essential part of the system, but it does not do all the work; there are still other steps toward detecting the occupancy of parking spaces.
As the first step, our system takes the input video, then reads the video frame by frame and does occupancy detection for each frame separately.

After reading the frame and detecting all the cars in the frame, we need the coordinates of each parking space. For this purpose, we used a script from an article on Medium.com [23]. The author provides a script that can be used to save the coordinates of selected polygons in a pickle file. We took this script and modified it enough to make it appropriate for our case. The modified version takes as an argument the path of the video and saves the selected parking space coordinates in a numpy file as the output. After saving the parking space coordinates for our gas station dataset, we can measure the Intersection over Union (IoU) for every pair of parking spaces and bounding boxes of detected cars.

Intersection over Union (IoU) is an evaluation metric used in object detection to evaluate the accuracy of the detected object's bounding box. It is the ratio of the intersection area between the ground truth bounding box and the predicted bounding box to the union area of both boxes. In this case, we use this measure to decide whether the parking space is occupied. Here it measures the ratio of the intersection area between the parking space and the car to the union area of both (Figure 5).

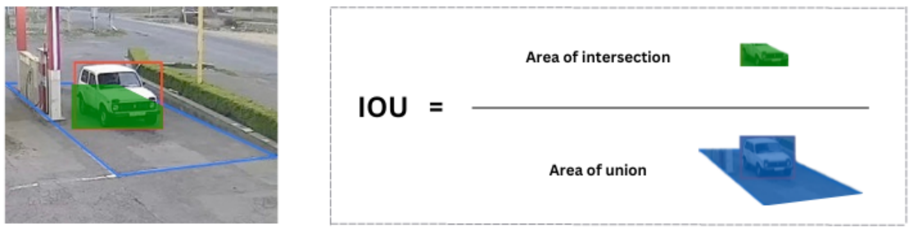


Figure 5. Real-time situation of the (a) intersection area, and (b) IoU.

We save the IoU values in an M × N matrix, where M is the number of parking spaces, and N is the number of detected cars. Then for each parking space, we take the maximum IoU value that it has with a car and compare that value with the threshold we set. If the IoU is greater than the threshold, we consider the parking space to be occupied; otherwise, we consider it to be vacant.

The system displays the occupancy detection results and also saves them. To indicate the vacant parking spaces, the system colors them in blue. In Figure 6, you can see what the output of the system looks like.

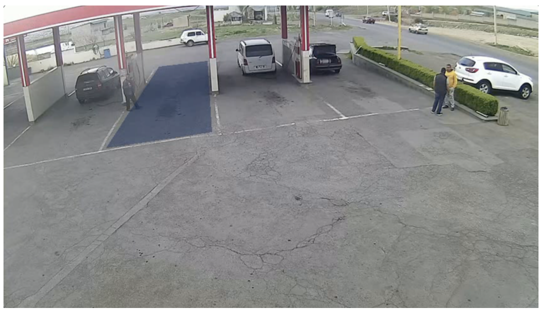


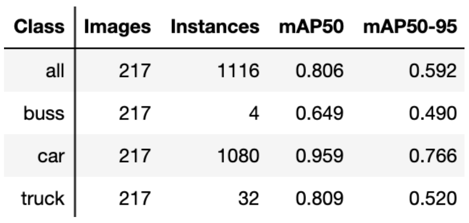
Figure 6. The output of our real-time parking occupancy detection system

## 6. Results and Discussion

In this section, we discuss the performance of the proposed system and the selection criteria for the important parameters.

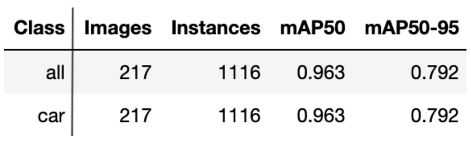
First, we should discuss the training process of the object detection model. For this task, we initially considered two approaches: training the YOLOv8 model as a single-class or multiclass model. Here the single-class model detects all three types of vehicles in a single class, whereas the multiclass model differentiates the types of vehicles, considering them as instances of separate classes. We started with training the multiclass model, and noticing that the results were not good, we also trained the single-class model. For the multiclass model, the mAP values were low for the bus and truck classes; for the car class, it was comparably higher (Figure 7a). We tested this model on the videos in the test dataset and noticed that the model could not detect buses, and they were often detected as cars or trucks (Figure 8a). The reason for these results is that in our dataset, we have only a few images with a bus and a small number of images that include a truck. Because of the lack of instances from truck and bus classes in the training dataset, the model was not able to learn all the features of these objects and, therefore, could not detect them accurately. If we had a larger dataset with more images of buses and trucks, it would possibly solve this problem.

To get a better object detection model, we trained the YOLOv8 as a single-class model. This model detects cars, trucks, and buses, but all of them belong to the same class named car. In this case, we got around 15% higher mAP50 than the single-class model, having a mAP50 of 0.806 for the multi-class model and 0.963 for the single-class model. Training the model as a single class would not cause any problems for parking occupancy detection because when we want to understand if the parking space is vacant, it is not important whether the parked vehicle is a car, truck, or bus; in any case, it will be considered as occupied.



| Class | Images | Instances | mAP50 | mAP50-95 |
|---|---|---|---|---|
| all | 217 | 1116 | 0.806 | 0.592 |
| buss | 217 | 4 | 0.649 | 0.490 |
| car | 217 | 1080 | 0.959 | 0.766 |
| truck | 217 | 32 | 0.809 | 0.520 |

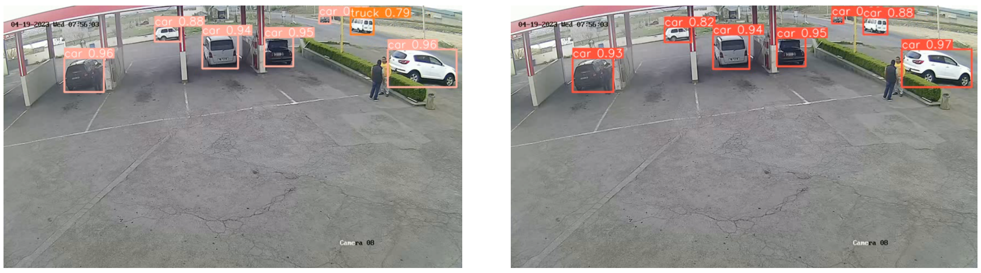
(a)



| Class | Images | Instances | mAP50 | mAP50-95 |
|---|---|---|---|---|
| all | 217 | 1116 | 0.963 | 0.792 |
| car | 217 | 1116 | 0.963 | 0.792 |

(b)

Figure 7. Evaluation of the trained (a) multi-class and (b) single-class models on the test dataset.

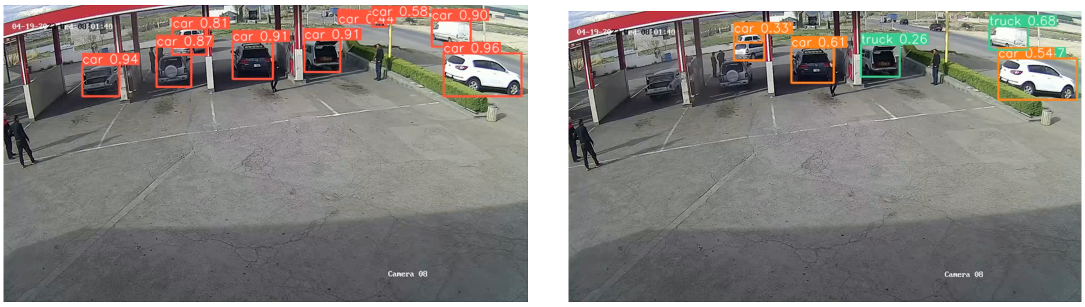


(a) (b)

Figure 8. Object detection results of trained (a) multi-class and (b) single-class YOLOv8 models.

After doing these comparisons, we concluded that the single-class model is a better option for parking occupancy detection and used it in the development of our system.

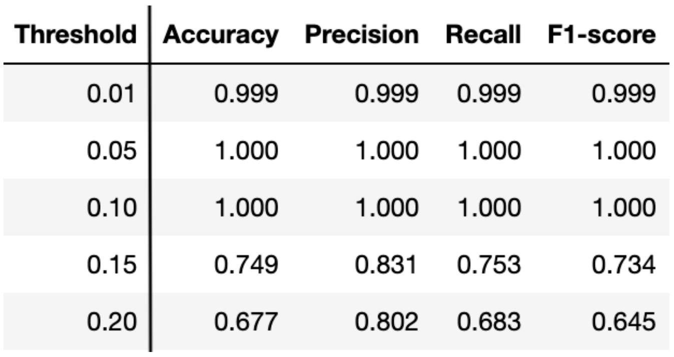
After training our model, we also compared it with the YOLOv8 model pre-trained on the COCO dataset. We performed object detection on our test dataset using both our trained model and the official pre-trained model. After comparing the results, we noticed that our model detects the cars better than the pre-trained model. The reason for that difference is the dataset. First of all, the training and test datasets for our model are similar; they are different parts of the same dataset, while the training dataset of the simple pre-trained model is the COCO dataset, which is very different from the test dataset we used here. Additionally, we noticed the model trained on the COCO dataset can not detect cars with an opened trunk (Figure 9). We think that the reason for this is the small number of images that have cars with opened trunks in the COCO dataset on which the model was trained. In contrast, our model does not have this problem. In our training dataset, we have many cars with open trunks and because of this, our model is able to detect cars with both open and closed trunks.



(a) (b)

Figure 9. Object detection results of (a) our trained model and (b) the YOLO v8 pre-trained model.

After having an accurate object detection model, we should also check the accuracy of the final parking occupancy detection model. The remaining problem consequently becomes a classification task because here we know the location of the parking spaces in the video, and we merely want to understand whether it is occupied or not. To be able to test the parking occupancy detection model, we labeled our test datasets' parking spaces as occupied or not, and after doing detection using our model, we compared the ground truth parking spaces' classes and the detected results by measuring the accuracy, precision, recall, and F1-score. We performed this several times for different thresholds of IoU for the parking space and a car, and we chose the threshold that gives the highest accuracy. From Figure 10, we can see that both 0.05 and 0.1 for the threshold give similar results. But we chose 0.1 because choosing 0.05 could potentially cause false positive values in the future. The parking space will be considered occupied also for the case where there is a car nearby but not parked in the space. We can see this in Figure 10 for a threshold of 0.01, where the accuracy decreased because of the false positive in the prediction. We can also notice that taking a larger value for the threshold decreases the accuracy even more. This is because, in case of a larger threshold, the parking space will be considered vacant also, for the case where there is a car parked, but it is in the corner of the parking space, and it does not cover a large part of it. Increasing the threshold even more, will not make sense as it will cause an even larger number of false negatives in the detection.



| Threshold | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| 0.01 | 0.999 | 0.999 | 0.999 | 0.999 |
| 0.05 | 1.000 | 1.000 | 1.000 | 1.000 |
| 0.10 | 1.000 | 1.000 | 1.000 | 1.000 |
| 0.15 | 0.749 | 0.831 | 0.753 | 0.734 |
| 0.20 | 0.677 | 0.802 | 0.683 | 0.645 |

Figure 10. Accuracy, precision, recall, and F1-score values for or test dataset for different thresholds of IoU between the parking space and a car

## 7. Conclusions and Future Work

In this project, we used deep learning to develop a real-time parking occupancy detection system. The suggested system consists of two components: object detection using YOLO and parking space occupancy detection using IoU. After testing on our test dataset, we saw that for detecting parking space occupancy, our system had high accuracy for the chosen threshold of IoU. For the object detection model, we also got good results, having an mAP value equal to 0.96. This shows that our system works with high accuracy and can be applied in real-life situations.

The parking occupancy detection system that we have developed can be further improved to get higher accuracy and efficiency. One step towards improving the system is the collection of more data for fine-tuning the object detection model. Another method is the usage of semantic segmentation instead of using object detection. In this case, we can calculate the IoU between the parking spaces and the masks of the detected car, and the resulting IoU will be a better measure for detecting occupancy. Another aspect of the parking occupancy detection system that has been left out of this capstone project is the development of a user interface to make it more accessible and user-friendly.

## References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *arXiv.org*, 09-May-2016. [Online]. Available: https://arxiv.org/abs/1506.02640. [Accessed: 05-May-2023].

[2] D. Acharya, W. Yan, and K. Khoshelham, “Real-time image-based parking occupancy detection using Deep Learning,” Apr-2018. [Online]. Available: https://www.researchgate.net/publication/323796590_Real-time_image-based_parking_occupancy_detection_using_deep_learning. [Accessed: 04-May-2023].

[3] X. Ding and R. Yang, “Vehicle and parking space detection based on improved Yolo Network Model,” 2019. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1325/1/012084. [Accessed: 04-May-2023].

[4] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv.org*, 08-Apr-2018. [Online]. Available: https://arxiv.org/abs/1804.02767. [Accessed: 05-May-2023].

[5] L.-C. Chen, R.-K. Sheu, W.-Y. Peng, J.-H. Wu, and C.-H. Tseng, “Video-based parking occupancy detection for smart control system,” *MDPI*, 06-Feb-2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/3/1079. [Accessed: 05-May-2023].

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” *arXiv.org*, 21-Mar-2019. [Online]. Available: https://arxiv.org/abs/1801.04381. [Accessed: 05-May-2023].

[7] “CNRPark-EXT,” *OpenDataLab*, 2016. [Online]. Available: https://opendatalab.com/CNRPark-EXT. [Accessed: 05-May-2023].

[8] A. A. Naufal1, C. Fatichah, and N. Suciati1, “Preprocessed mask RCNN for parking space detection in smart ... - INASS,” 2020. [Online]. Available: http://www.inass.org/2020/2020123123.pdf. [Accessed: 04-May-2023].

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” *arXiv.org*, 24-Jan-2018. [Online]. Available: https://arxiv.org/abs/1703.06870. [Accessed: 05-May-2023].

[10] T. Mertens, J. Kautz, and F. V. Reeth, “Exposure fusion - Stanford University.” [Online]. Available: https://web.stanford.edu/class/cs231m/project-1/exposure-fusion.pdf. [Accessed: 05-May-2023].

[11] S. Rahman, M. Ramli, F. Arnia, and R. Muharar, “Performance analysis of malexnet by training option and activation ...” [Online]. Available: https://www.researchgate.net/publication/349496636_Performance_analysis_of_mAlexnet_by_training_option_and_activation_function_tuning_on_parking_images. [Accessed: 05-May-2023].

[12] Z. S. Scekic, S. Cakic, and T. Popovic, “Image-based parking occupancy detection using Deep Learning and faster ...,” 2022. [Online]. Available: https://www.researchgate.net/profile/Tomo-Popovic/publication/358734655_Image-Based_Parking_Occupancy_Detection_Using_Deep_Learning_and_Faster_R-CNN/links/6291a0e66886635d5ca889a2/Image-Based-Parking-Occupancy-Detection-Using-Deep-Learning-and-Faster-R-CNN.pdf. [Accessed: 04-May-2023].

[13] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” *arXiv.org*, 06-Jan-2016. [Online]. Available: https://arxiv.org/abs/1506.01497. [Accessed: 05-May-2023].

[14] Facebookresearch, “Facebookresearch/Detectron2,” *GitHub*. [Online]. Available: https://github.com/facebookresearch/detectron2. [Accessed: 05-May-2023].

[15] P. Almeida, L. S. Oliveira, A. de S. Britto Jr, and E. J. Silva Jr, “PKLOT - a robust dataset for parking lot classification - researchgate,” 2016. [Online]. Available: https://www.researchgate.net/publication/273479425_PKLot_-_A_Robust_Dataset_for_Parking_Lot_Classification. [Accessed: 05-May-2023].

[16] *Google colab*. [Online]. Available: https://colab.research.google.com/. [Accessed: 05-May-2023].

[17] *NumPy*. [Online]. Available: https://numpy.org/. [Accessed: 05-May-2023].

[18] *OpenCV*. [Online]. Available: https://opencv.org/. [Accessed: 05-May-2023].

[19] “Revolutionizing the world of Vision Ai,” *Ultralytics*. [Online]. Available: https://ultralytics.com/. [Accessed: 05-May-2023].

[20] “Give your software the power to see objects in images and video,” *Roboflow*. [Online]. Available: https://roboflow.com/. [Accessed: 05-May-2023].

[21] Ultralytics, “Ultralytics/ultralytics,” *GitHub*, Jan-2023. [Online]. Available: https://github.com/ultralytics/ultralytics. [Accessed: 05-May-2023].

[22] “Common objects in context,” *COCO*. [Online]. Available: https://cocodataset.org/#home. [Accessed: 05-May-2023].

[23] M. Saini, “Parking space detection using deep learning,” *Medium*, 03-Dec-2019. [Online]. Available: https://medium.com/the-research-nest/parking-space-detection-using-deep-learning-9fc99a63875e. [Accessed: 05-May-2023].