

University of Southern Indiana
Pott College of Science, Engineering, and Education
Engineering Department
8600 University Boulevard
Evansville, Indiana 47712


Automated Parking Lot Monitoring System

Vision Learning System Using NVIDIA Jetson Nano

Adam Kirkham

ENGR 491 – Senior Design

Spring 2023

Approved by:  April 27, 2023

Faculty Advisor: Milad Rad, Ph.D.

Date

Approved by: _____

Department Chair: Paul Kuban, Ph.D.

Date

ACKNOWLEDGEMENTS

In this section, I would like to acknowledge Dr. Rad for his advising throughout this project, Dr. Kuban and the USI engineering department for the funding for this project, Kassidon Hatfield for his assistance and advising with the Python portions of this project, and Tim Desser for and Kendal Roeder for their advising on electrical aspects of this project. I would also like to acknowledge Dr. Field and Dr. Nelson for their efforts in academic advising that have led to my graduation this semester.

ABSTRACT

The purpose of this project was to create a proof-of-concept vision based automated parking space monitoring system to monitor the number of empty parking spaces in a defined area. This project was specifically targeted toward parking lot J at the University of Southern Indiana with a desired camera placement on the third floor of the Business and Engineering Center. This lot was selected as it has limited spaces while being the closest lot to several university buildings, resulting in high vehicle and pedestrian traffic during peak times. The original idea to solve this proposed by various faculty was to implement a camera and image processing solution in a third-floor window of the USI Business and Engineering building as it would be possible to do so with no added infrastructure. A computer vision model using the opensource program TensorFlow was selected as a solution to this problem due to the relatively low cost and minimal necessary infrastructure for operation. This software was implemented on an NVIDIA Jetson Nano using the SSD Mobilenet detection architecture. A transfer learning approach was used to retrain the SSD Mobilenet detection architecture to tailor it to this application. Custom datasets were generated to improve results. An additional detection model was created to automate parking space location detection. The final design is able to accurately detect vehicles within a designated area. Due to limitations regarding perspective and image processing resolution, the final design cannot be implemented solely through camera placement on the third floor of the Business and Engineering building. A cost analysis determined that the added infrastructure required to implement a multi-camera solution would be similar to the cost of traditional sensor-based solutions.

TABLE OF CONTENTS

Acknowledgements	i
Abstract.....	ii
List of Figures.....	iv
List of Tables	vi
Acronyms	vii
1 Introduction.....	1
1.1 Objective	1
1.2 Deliverables.....	1
1.3 Project Motivation.....	1
1.4 Factors Impacting Design.....	3
1.5 Literature Review	3
2 Methodology	8
2.1 Computer Vision Selection	8
2.2 System Structure and Theory	9
2.3 Software Needed	9
2.4 SSD Mobilenet	11
2.5 Multi-Camera Implementation – Initial Model Calibration	15
3 System Design.....	17
3.1 Potential Camera Positions.....	17
3.2 Detection Results and Transfer Learning.....	28
3.3 Cost Analysis	31
4 Conclusions and Recommendations.....	34
4.1 Project Conclusion	34
4.2 Future Recommendations.....	34
References	36
Appendix.....	39

LIST OF FIGURES

Figure 1. Aerial View of Parking Lot J [1].....	2
Figure 2. Parking Lot J Light Post Positions [1].....	2
Figure 3. intuVision Parking Lot Demo. Adapted from: [7]	5
Figure 4. Outdoor Parking Space Sign, Sign-Tech, [9]	6
Figure 5. NVIDIA Jetson Nano Developer Kit, NVIDIA, [15].....	8
Figure 6. CUDA Installation Target Guide. [17].....	11
Figure 7. Example Object Detection with NVIDIA Jetson Nano and SSD Mobilenet [16]	12
Figure 8. Example Training Output to Terminal	13
Figure 9. Label Studio Example Custom Image from Dataset	15
Figure 10. Maximum Required Data Transmission. Adapted from [1].....	16
Figure 11. Effect of FOV on Number of Visible Parking Spaces	17
Figure 12. Proposed Camera Placement Location on B.E. Center. Adapted from [1]	18
Figure 13. Camera View from Southwest BE Third Floor Stairwell	19
Figure 14. Spaces with Potentially Obstructed Views.....	20
Figure 15. Camera Locations with Added Ceramic Center Camera. Adapted from [1]	21
Figure 16. Alternative Perspective on Obstructed Spaces	21
Figure 17. Vehicle Detection with Overlapping Cars.....	22
Figure 18. Camera Coverage Map with Light Post Locations.....	23
Figure 19. Furthest light post from Business and Engineering Center. Adapted from [1]	24
Figure 20. Effects of Image Compression on Smaller Detection Area.....	26
Figure 21. Effects of Image Compression on Smaller Detection Area.....	26
Figure 22. Effect of Detection Area on Detection Results	27
Figure 23. Example Missed and Incorrect Detections with SSD-Mobilenet.....	28

Figure 24. SSD-Mobilenet Detection Results – Retrained with Open Images Datasets 29

Figure 25. Typical Views of Car in Open Images Dataset. [23]..... 30

Figure 26. Detection Error Improvement with Additional Training Iterations 31

LIST OF TABLES

Table 1. Summary of System Costs	32
Table 2. Cost of Per-Space Sensor Implementation [23] [24]	32
Table 3. Cost of Ingress-Egress Sensor Implementation [23] [24] [25]	32
Table 4. Cost of Multi-Camera Detection Implementation [15].....	32
Table 5. Cost of Hybrid Ingress-Egress Ultrasonic and Detection System [15] [23].....	33

ACRONYMS

FOV – Field of View

B.E. (building) – Business and Engineering Building

DNN – Deep Neural Network

GUI – Graphical User Interface

USI - University of Southern Indiana

1 INTRODUCTION

The monitoring of automobile traffic, in this case within a parking lot, proposes a multidisciplinary engineering design problem. Due to space limitations, parking lots in high traffic areas often fill rapidly during peak use times. On the University of Southern Indiana Campus, parking lot J adjacent to the Business and Engineering Center experiences high traffic before regularly scheduled class times. Students and faculty often circle the lot looking for a space, resulting in wasted time before parking in the nearest alternate lot, lot F. The time taken to search lot J for a space in conjunction with the time taken to walk approximately 300 feet from lot F leads to frustration and tardiness for students and faculty. This report discusses several design solutions with an in-depth discussion regarding the development of computer vision models to monitor the availability of open parking spaces in a specified area.

1.1 OBJECTIVE

The objective of this project was to develop an image-processing based parking lot monitoring system and evaluate the feasibility for application of such a system in parking lot J on the University of Southern Indiana campus.

1.2 DELIVERABLES

The scope of this project included the design and development of a proof-of-concept prototype system. Due to time and budgetary restrictions, a full installation was out of the scope of this project. The final design must be able to identify the number of empty spaces in a defined area of a parking lot and output the results in real time.

1.3 PROJECT MOTIVATION

This project was driven by the inconvenience and safety hazard created by the limited parking available in parking lot J on the University of Southern Indiana campus. The lot in question has 205 parking spaces with 17 of these being handicap accessible parking. The 188 available non-handicap parking spaces do not meet the daily need within this lot. This leads to cars circling the lot in search of open spaces during peak foot traffic times, mainly the 10-minute passing periods between regularly scheduled lectures. In most of these cases, the lot is already full, leading to drivers wasting time before having to park in an alternate lot. Figure 1 below is an aerial view of the lot with the individual parking spaces numbered. A larger version of this image can be found

in appendix A. The image was adapted from the Google maps “Terrain” layer accessed in January 2023.



Figure 1. Aerial View of Parking Lot J [1]



X – INDICATES LIGHT POST LOCATIONS

Figure 2. Parking Lot J Light Post Positions [1]

1.4 FACTORS IMPACTING DESIGN

The system design needs to comply with certain design factors. The primary design concern was budget. While such monitoring systems are not a new concept, the goal of this project was to design a low budget solution that could be easily implemented internally by the university. Because of this, it was necessary to minimize the amount of infrastructure required to implement the final design.

Existing infrastructure surrounding the parking lot includes 14 light posts, the USI Business and Engineering (B.E.) building, the USI Art Studio Building, and the USI Ceramic Center. The building locations can be seen above in Figure 1, and the light post positions can be seen above in Figure 2. The light post positions indicate the only current outdoor electrical infrastructure. The lights in question are powered by three phase power according to the USI Facility Operations and Planning worker Kendal Roeder. This power source would not be compatible with devices designed to work with standard US receptacles.

1.5 LITERATURE REVIEW

1.5.1 Existing Solutions

Parking lot management systems are not a new concept. Systems in which the number of parking spaces available in a given parking lot or parking garage are becoming more common in areas of high traffic to increase the efficiency of parking space use [2]. As the motivation for this project involves a public accessible parking lot, monitoring and regulation systems based on gated access, parking passes, paid parking, or RFID monitoring are considered not applicable for this project.

Current parking space monitoring systems work through detecting the presence of a vehicle through one or more of the following methods. In most systems, vehicle detection is achieved through the implementation of ultrasonic sensors, induction coils, image processing, or RFID or similar sensors [3]. Additional systems use magnetic sensors to detect the ferrous metals commonly found on vehicles. Despite the sensor type, all systems utilizing sensor-based detections function by monitoring ingress and egress points of the lot or by monitoring individual parking spaces for the presence of a vehicle [2]. Such sensor-based solutions have a potential for

inaccurate counts due to non-vehicles being incorrectly detected as vehicles. Such inaccurate detections include but are not limited to ultrasonic sensors detecting the presence of a person or other object as a vehicle, or induction and magnet-based sensors detecting metallic, non-vehicle objects as a vehicle [3]. While ultrasonic sensors are exposed to the elements, induction-based systems have the advantage of requiring virtually no maintenance due to their underground installation. Ultrasonic and similar type sensors have the advantage of having more versatility in where they are mounted [4].

Recent developments in parking management systems have seen a shift toward image processing solutions. Such systems attempt to solve the previously mentioned detection issues through intelligent machine learning vision systems. Such systems are also desirable due to the versatility in implementation around existing infrastructure [5]. Such systems are available commercially through companies such as Chetu or intuVision. Both of these companies are technology companies with diverse products and services including various machine vision applications. Chetu's offerings focus on software-based monitoring and analytics with a focus on private and restricted access parking [6]. Similarly, intuVision provides parking space, parking lot, and parking garage monitoring systems with emphasis on data analytics. Space count monitoring systems by intuVision offer ingress/egress-based detections or individual space monitoring [7]. Figure 3 below showcases the detection capabilities of intuVision monitoring systems.



Figure 3. intuVision Parking Lot Demo. Adapted from: [7]

1.5.2 Driver Communications

Multiple interfaces exist for communicating current space counts to drivers. Most systems utilize electronic signs at parking lot or garage ingress points to display current occupancy. Such signs are widely commercially available through various manufacturers. Alternative solutions are available through various web-based and smartphone applications [8]. An example of an electronic occupancy sign at the entrance of a parking lot is seen in Figure 4 below.



Figure 4. Outdoor Parking Space Sign, Sign-Tech, [9]

1.5.3 Imaging Processing

The use of image processing to monitor parking spaces is not a novel idea, as can be seen by the commercial products discussed above. In an effort to avoid the infrastructure changes associated with mounting multiple cameras in lot J, several other strategies were investigated. The use of drones to capture camera footage was one option that could solve this problem. Such systems have been investigated for use monitoring street parking in busy cities such as is discussed in [10]. This solution would not provide accurate counts during high traffic due to the time taken to scan the area. Because of this, this solution was considered to not be applicable to this project.

While most image processing solutions use a detection based DNN for the locating of vehicles, this solution does not offer accurate locating of parking spaces. An alternative detection network investigated was the use of semantic segmentation. This detection strategy classifies object classes in an image on a pixel-by-pixel basis. This style of application has been applied to similar applications as is discussed in [11] and [12]. While this solution would provide easier detection of parking spaces, it is not useful for providing quantities of objects detected. Because of this, this application was not considered applicable to this project.

The two primary detection networks used for vehicle detection are YOLO (You Only Look Once) and SSD Mobilenet. These two options offer similar detection results by applying similar image processing techniques. YOLO is capable of detecting smaller objects within an image, though it also requires greater processing power to do so. SSD Mobilenet is optimized for mobile applications and thus is able to run on devices with lower processing power [13].

The base principles of detection are shared between the two aforementioned detection networks. The DNN is trained through inputting tens of thousands of images with object classes and bounding boxes labeled. Images input into the network are compressed to a standard image size and converted to black and white. The DNN compares contour lines within the defined bounding boxes to determine common characteristics between the images in the object class. To detect objects, the detection network applies the same compression and black and white conversion to the input image and searches for known contour lines. If known contour lines are detected, the network will output the estimated bounding box, object class, and confidence to the user [13].

1.5.4 Environmental Impacts

By reducing the number of vehicles checking an already full parking lot, unnecessary fuel consumption is reduced. Reducing fuel consumption not only reduces use of a nonrenewable resource but also reduces greenhouse gas emissions associated with combustion engine vehicles [14].

2 METHODOLOGY

2.1 COMPUTER VISION SELECTION

For this project, computer vision was achieved using TensorRT in conjunction with the NVIDIA Jetson Inference deep neural network (DNN) libraries and development tools. This was run on an NVIDIA Jetson Nano Developer Kit, pictured below in Figure 5, a small Linux based computer made for running neural networks and computer vision models. The Jetson Nano was chosen due to its processing capabilities and small formfactor. Alternative similar choices for this project would include the Raspberry-pi or similar single board computers. Use of a board other than the Jetson Nano would require different detection methods as the methods used in this project are made only to work with NVIDIA graphics cards.



Figure 5. NVIDIA Jetson Nano Developer Kit, NVIDIA, [15]

2.2 *SYSTEM STRUCTURE AND THEORY*

The main goal of the detection model was to determine the number of empty parking spaces available in a given area. Several methodologies were considered for solving this solution. All of these initially considered solutions were based on object detection utilizing the Mobilenet SSD pre-trained neural network and the detectNet program by Nvidia.

The first methodology would be to train the machine vision model to detect empty parking spaces. This would require the model to be able to identify a parking space location and determine if a car was present in the space. This could be done by training the model to detect parking spaces and cars, or specifically to detect empty parking spaces. Due to the similarity between the parking spaces and the driving lanes between the spaces, it was determined that following this solution could produce inaccurate results.

Alternatively, the machine vision model could be trained to identify the locations of cars and check this against the location of known parking spaces. This would require the model to either identify the location of each parking space or have the locations manually input into the program. As was previously discussed, locating the specific parking spaces was prone to inaccuracies. Manually inputting the space locations would require tedious user input and would also be prone to inaccuracies from camera movement.

The final option that was pursued for this project was to only train the model to detect the locations of cars and handicapped-accessible parking spaces. Due to the consistent nature of handicapped-accessible parking space identifiers, the model was able to be trained to detect the location of such spaces. Additionally, due to the commonality of car object detection, the training and use of such detections is proven and well documented. With this strategy, the locations of handicapped-accessible spaces would be located and documented with the spaces empty. The model would then only need to monitor the number of detected vehicles and note when vehicles are in handicapped-accessible spaces to keep an accurate count of empty standard and handicapped-accessible spaces.

2.3 *SOFTWARE NEEDED*

Several software packages are needed to complete the work as documented in this report. This section will cover the software used in this project and how to install said software.

2.3.1 Jetson Nano Setup

The Jetson Nano used for this project must initially be set up following the *Getting Started with Jetson Nano Developer Kit* tutorial available through the NVIDIA Developer website. In this tutorial, the Jetpack Linux based operating system and necessary basic programs will be installed [15].

To set up the Jetson Nano Developer Kit for image processing, several libraries are needed. The process for this is well documented in the NVIDIA jetson-inference GitHub, but several key steps are outlined below. Most necessary software is included with the NVIDIA *Hello AI World* inferencing library. This is most simply done through the jetson-inference docker container. This container can be installed using the following commands in the command window of the Jetson Nano [16].

```
$ git clone --recursive https://github.com/dusty-nv/jetson-inference
```

```
$ cd jetson-inference
```

```
$ docker/run.sh
```

Running the docker container will automatically begin the installation of all NVIDIA inferencing libraries needed for this project. For the creation of custom models, PyTorch should be installed when prompted. PyTorch for Python 3.6 should be selected.

2.3.2 Separate Computer Setup

As was previously mentioned, the Jetson Nano runs on a Linux operating system. In order to speed up training processes, a separate more powerful computer is recommended. A computer with an NVIDIA graphics card is preferred as NVIDIA CUDA only works with NVIDIA graphics cards. If the computer being used does not operate on a Linux operating system, a parallel operating system must be used. This is most easily done by installing Ubuntu,

a free Linux operating system. Ubuntu can be run as a program within Windows and is available for free through the Microsoft Appstore. More information on this topic is available through the Ubuntu tutorials webpage.

In order to run the necessary training programs on a separate computer, NVIDIA CUDA and cuDNN are required. CUDA can be installed through directions on the NVIDIA Developer website downloads page [17]. Directions for the installation of CUDA will be generated on the page based on information regarding the machine running CUDA. The Linux version of CUDA should be installed as it will be run within Ubuntu. The “Version” selected should match the version of Ubuntu installed previously. Figure 6 Below shows the configuration installed for this project running Ubuntu within Windows. Commands will then be generated in the “Download Installer” box below “Select Target Platform” box. These commands should be copied and pasted into Ubuntu one at a time and run to install CUDA.

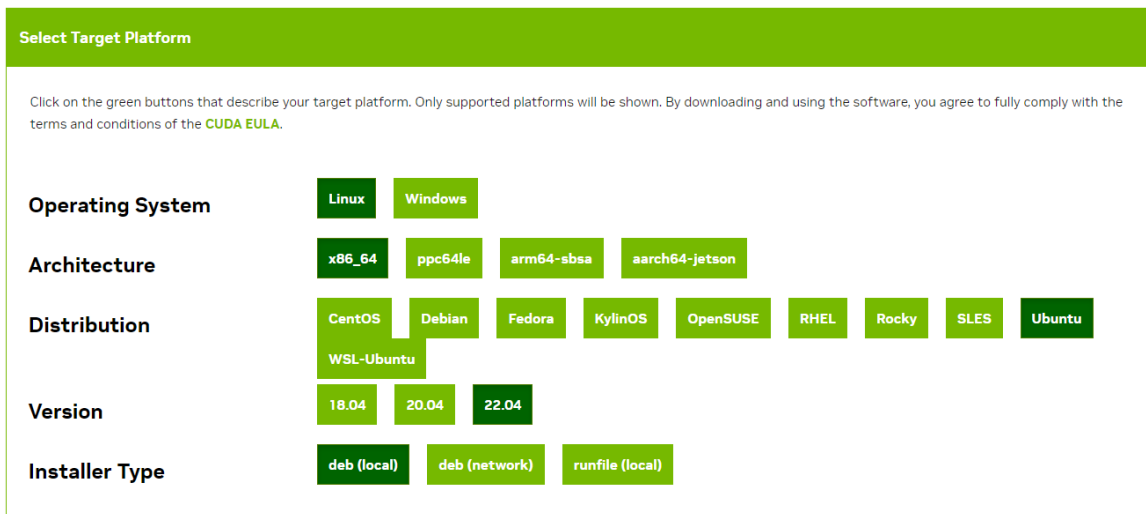


Figure 6. CUDA Installation Target Guide. [17]

Once CUDA is installed, cuDNN can be installed following the directions on NVIDIA cuDNN webpage. It is critical to select the installer for the version of Ubuntu previously downloaded [18].

2.4 SSD MOBILENET

2.4.1 Overview

The image processing was achieved using a Single-Shot multibox Detection (SSD) network known as SSD Mobilenet. This pre-trained detection model was chosen due to its capability to

scan entire images and detect multiple objects and wide array of pre-trained object classes. The SSD network structure provides faster processing as compared to other detection networks that scan individual pixels or segments of an image (Choudhari et al.). This was implemented using the Nvidia detectNet program as this allows for object detection in still images or live camera feeds and is optimized for use with the Nvidia Jetson Nano. Image data was extracted from the detectNet using the Nvidia jetson.inference python library. The SSD Mobilenet detection architecture is pre-trained to detect 91 classes of objects. The classes included that were of interest for this project were person, bicycle, car, motorcycle, bus, and truck. Figure 7 shows an example detection achieved running this model on the NVIDIA Jetson Nano [16].

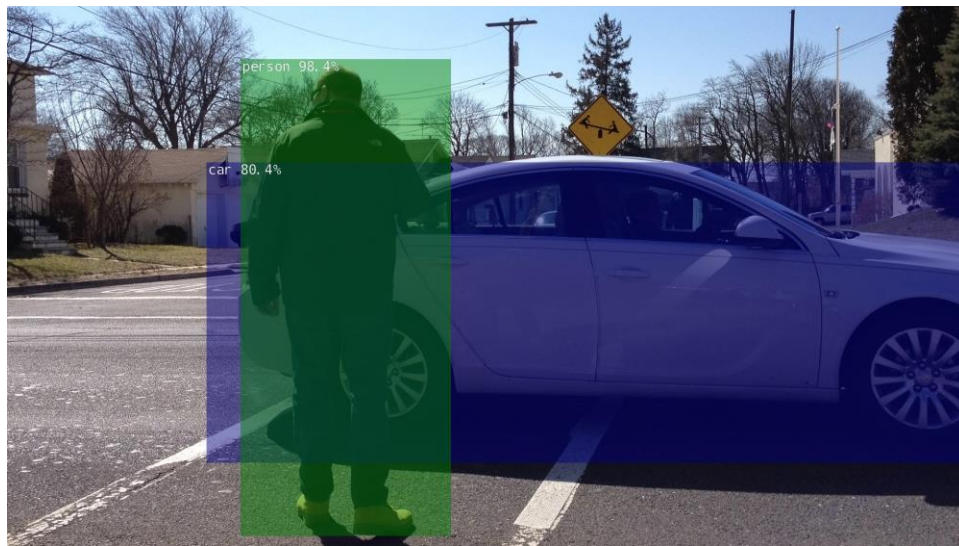


Figure 7. Example Object Detection with NVIDIA Jetson Nano and SSD Mobilenet [16]

2.4.2 Model Training with Pytorch

As will be discussed further in the initial results section on page 28, the SSD-Mobilenet model failed to provide accurate detection results during initial testing. To improve the performance of the detection architecture, the model was retrained using the transfer learning technique. Transfer learning allows for the creation of a custom neural network through the manipulation of an existing framework. This method was especially effective for this application as the SSD Mobilenet architecture was already trained to detect vehicles, so allowing for Inductive Transfer of the neural network, or a simple reduction in the number of classes the infrastructure searches for [19]. While a custom-made detection network could potentially

generate better results, the work involved with developing a custom neural network was determined to be outside the scope of this project. The opensource machine learning framework PyTorch was applied using the NVIDIA program train_ssd to complete the transfer learning process.

Initial transfer learning attempts were completed using the Open Images opensource computer vision dataset from OpenCV. Datasets for the classes bicycle, bus, car, person, truck, and van were used. Further training was conducted using custom datasets created using the National Instruments Label Studio dataset creation software. Photos were taken from the second and third floor of the southwest stairwell in the B.E. Center. Photos were then selectively cropped to create a dataset containing vehicles of different makes, models, colors, and perspectives. Bounding box data was then added using National Instruments Label Studio.

Transfer learning retraining of SSD Mobilenet was conducted using the train_ssd program installed as part of the jetson-inference libraries. This program retrained the SSD Mobilenet detection network utilizing the following inputs:

- Dataset directory
- Model directory
- Number of epochs to run (training iterations)
- Photo batch size (affects processing time and required processing power)
- Number of workers (affects processing time and required processing power)

After ten completed dataset batches, or groups of training photos, the program outputs the average regression and classification loss results to the terminal. Regression loss refers to accuracy of bounding box placement, and classification loss refers to class identification errors. Figure 8 below shows an example set of terminal outputs. The example below is from a training with a dataset of 44180 photos and a batch size of 10. This is reflected in the “Step” output of 4418. The average loss output is a general indicator of the network performance. It is desirable to increase the number of training epochs until this value is minimized.

```
2023-03-13 02:47:02 - Epoch: 0, Step: 1530/4418, Avg Loss: 3.5685, Avg Regression Loss 1.2928, Avg Classification Loss: 2.2757
2023-03-13 02:47:12 - Epoch: 0, Step: 1540/4418, Avg Loss: 3.7965, Avg Regression Loss 1.4563, Avg Classification Loss: 2.3403
2023-03-13 02:47:21 - Epoch: 0, Step: 1550/4418, Avg Loss: 3.6437, Avg Regression Loss 1.4047, Avg Classification Loss: 2.2390
2023-03-13 02:47:31 - Epoch: 0, Step: 1560/4418, Avg Loss: 3.4013, Avg Regression Loss 1.2594, Avg Classification Loss: 2.1419
2023-03-13 02:47:40 - Epoch: 0, Step: 1570/4418, Avg Loss: 3.7801, Avg Regression Loss 1.4747, Avg Classification Loss: 2.3055
```

Figure 8. Example Training Output to Terminal

Transfer learning through the use of `train_ssd` can be completed using Open-CV or VOC type datasets. Open-CV datasets are premade opensource datasets. VOC datasets are an alternate file format of dataset. VOC datasets contain less data. Because of this, the datasets may result in less accurate trained models. Due to the simplicity of the datasets, creation of these datasets is possible by the user. This can most easily be done through the use of the National Instruments program Label Studio or utilizing tools within the Jetson Inference detection libraries [16].

Transfer learning, while faster than the creation of a new detection network, takes a significant amount of time depending on the processing power of the computer, the dataset size, and the number of training iterations. For instance, the Jetson Nano is capable of processing 4.77 images per second [16]. For a dataset of 10,000 photos processed over 100 epochs (training iterations), this would require a minimum of 58 hours to process. As this example would be a considered a small dataset size and lower number of epochs, it can be seen how training can become a large time commitment. Because of this, a separate computer with greater processing power was used for trainings in this project. It is necessary for the computer to have an Nvidia graphics card for training using the Nvidia Jetson Inference libraries. Information regarding the software used is covered in greater detail above in the section above on separate computer setup. Despite the more powerful computer used, the total time spent running trainings in this project totaled over two weeks.

2.4.3 *Label Studio*

To create custom datasets in this project, Label Studio, a program made by National Instruments, was used. The program was used to generate object classes and bounding box data for custom images. Figure 9 below shows an example of the bounding boxes as would be drawn in Label studio. In the figure, the purple boxes denoted the object class “Truck” and the green boxes denoted the object class “Car”. This was completed for approximately 100 images of vehicles captured from the third floor of the B.E. building and cropped for varying relative vehicle size. This process was completed using a personal desktop running Ubuntu along with a internet browser based interface.

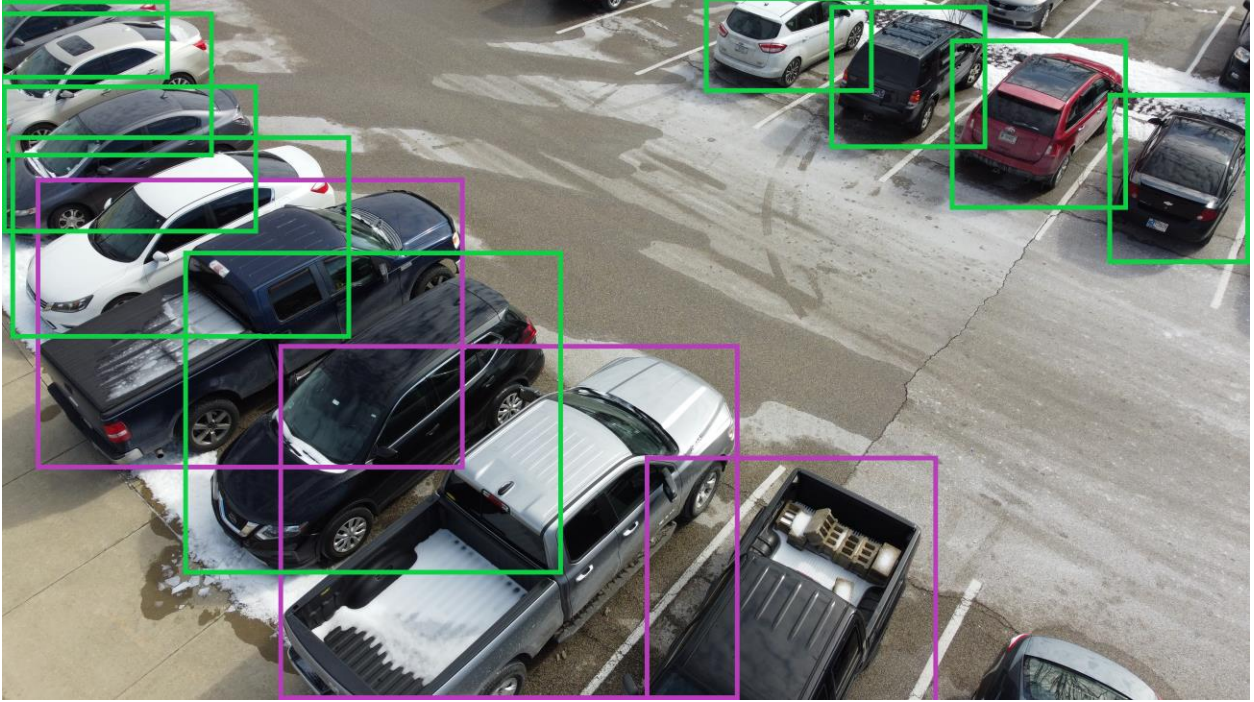


Figure 9. Label Studio Example Custom Image from Dataset

2.5 MULTI-CAMERA IMPLEMENTATION – INITIAL MODEL CALIBRATION

In order to monitor all areas of the parking lot without an obscured view of spaces, multiple camera locations would be necessary. The use of multiple cameras creates an issue regarding the accurate count of vehicles/parking spaces as camera views will overlap, causing vehicles in some spaces to be counted more than once. To solve this, the detection locations were filtered during data processing. The detection bounding areas for each camera could be manually programmed in, but this would require careful consideration as the pixel locations for the bounding boxes would be in terms of the 300×300 compressed image. To properly do this manually, a GUI would need to be created. It was determined that such a task is outside the scope of this project. An alternative solution to prevent overlapping camera views is to train a detection model for the purpose of an initial setup of the desired detection area. This was done utilizing the same transfer learning technique previously discussed. The model was retrained to only recognize the handicapped-accessible parking symbol, triangles, and circles. This was done such that a singular image can be processed to record bounding areas and handicapped-accessible parking spaces.

This detection model was trained using the same methods as were previously mentioned. The shapes used were black on a white background to maximize contrast. These shapes can be added by using physical large printouts of the shapes or by overlaying the shapes in a photo editing software. The coordinates of the bounding corners and locations of the handicapped-accessible spaces were saved to a .txt file to be read by the main detection python script.

Connecting multiple cameras also requires the transfer of data between devices. This would require either: a) Jetson Nano devices be mounted with each group of cameras and the resulting vehicle count be transmitted back to a central processing location, or b) the video feeds from the cameras be broadcast back to a centralized Jetson Nano for image processing. The data for either option could most easily be transmitted while remaining FCC compliant would be to transmit data over Wi-Fi. This would require a more powerful router to be implemented to create coverage within parking lot J. Assuming the data is centrally processed within the B.E. building, the longest distance the data would need to be transmitted is approximately 370 feet as can be seen measured with Google Maps below in Figure 10. This is within the ranges possible with outdoor Wi-Fi systems [20].



Figure 10. Maximum Required Data Transmission. Adapted from [1]

3 SYSTEM DESIGN

3.1 POTENTIAL CAMERA POSITIONS

3.1.1 Camera Field of View (FOV)

To provide input for a computer vision system, several camera positions were evaluated for feasibility. Evaluation of potential camera positions was conducted by taking photos from proposed perspectives using a 12 MP resolution and an 83° field of view (FOV) camera. To achieve elevated camera perspectives, images were captured from the B.E stairwell and cropped to approximate the perspective. As the FOV is different for different camera models, the FOV of the camera used in the final design may produce images differing from the photos captured. Figure 11 below shows how different the camera FOV would change the number of parking spaces visible to a camera mounted to a light post.

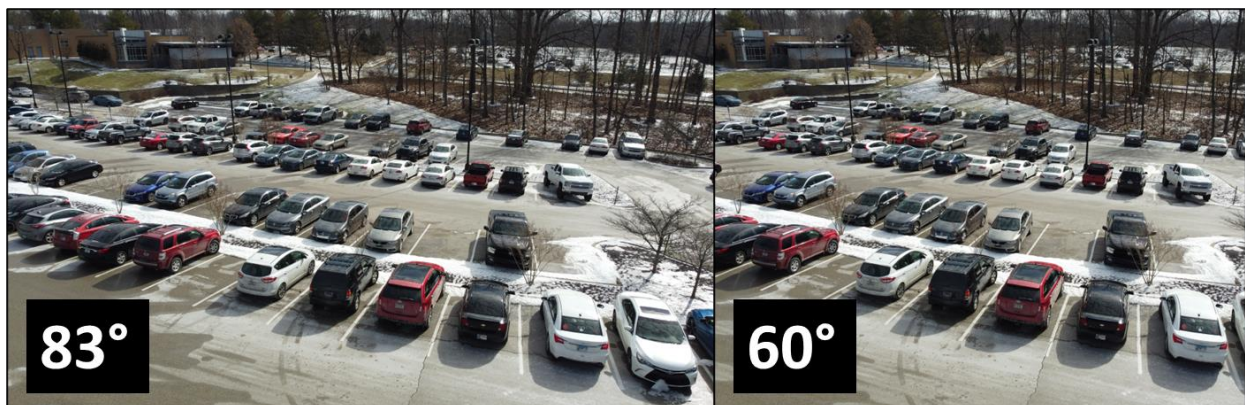


Figure 11. Effect of FOV on Number of Visible Parking Spaces

The FOV chosen for implementation not only changes the visible area, but more importantly the number of vehicles in a single image. As will be discussed in the image resolution considerations section on page 25, the size of a vehicle in the image directly affects the detection accuracy. While a wide-angle camera would reduce the number of necessary cameras for full area coverage, the decreased relative size of the cars in the image would result in poor detection performance.

3.1.2 Interior Camera Placement

The simplest camera position would be placement in a window of the B.E. building. Placing the camera inside would negate the need for any weatherproofing measures that would be necessary for an exterior installation. Placement in the 3rd floor window of the southwest stairwell would provide the highest elevation perspective while remaining out of reach to avoid undesired interference from students. Figure 12 shows the location of the potential camera position, while Figure 13 shows the view from this position.



Figure 12. Proposed Camera Placement Location on B.E. Center. Adapted from [1]



Figure 13. Camera View from Southwest BE Third Floor Stairwell

In Figure 13, it can be seen that the 83° FOV camera used could not capture the entire lot in one image. Analyzing the aerial perspective found in Figure 12 finds that the total angle of view needed is approximately 115° . Without using a wide-angle camera, a minimum of two cameras are needed for complete coverage. Due to factors that will be discussed later, more than two cameras would need to be installed for accurate detections.

The view in Figure 13 also proposes a secondary issue with this placement. Note that, due to the perspective of the camera, multiple vehicles are partially obstructed from view. This view obstruction is greatest with the parking spaces at the west side of the lot. These parking spaces pose the greatest risk for fully obscured vehicles. Figure 14 highlights these parking spaces. As will be discussed later in this report, obstruction of the view of a vehicle greatly reduces the accuracy of detection with the methods used in this project.



Figure 14. Spaces with Potentially Obstructed Views

While the proposed camera position in the B.E. building would not provide adequate coverage of the lot, the addition of a secondary camera on top of the USI Ceramic Center could potentially solve this. The location of the additional camera can be seen in Figure 15, and the view from this position can be seen in Figure 16. This camera position would provide a less obstructed view of the spaces highlighted in Figure 14. While this would provide additional information, the view could still be obstructed by large vehicles, leading to inaccurate readings. Figure 17 below shows how obstructed vehicle view can negatively affect detection results. In this example, the detection threshold was set to 95%.

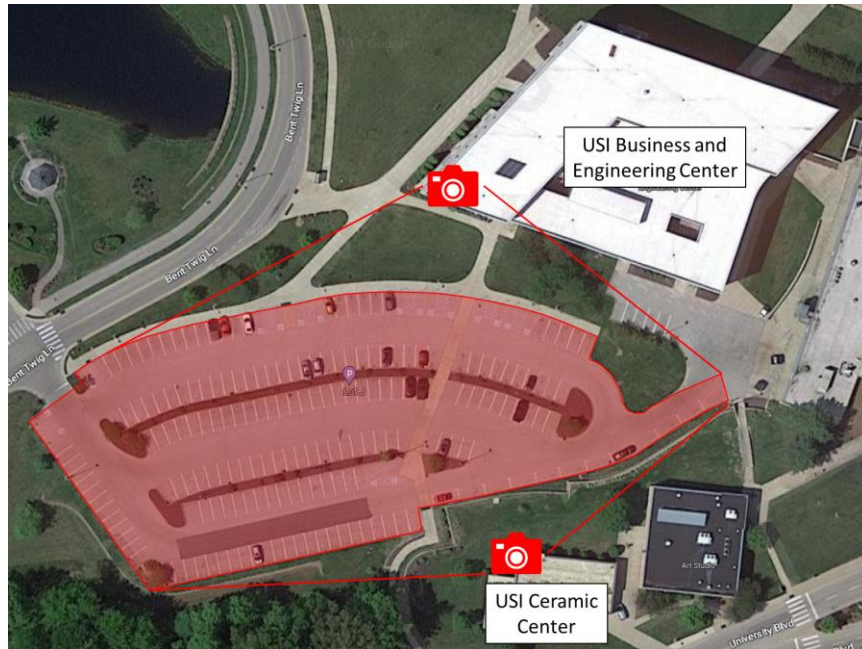


Figure 15. Camera Locations with Added Ceramic Center Camera. Adapted from [1]



Figure 16. Alternative Perspective on Obstructed Spaces

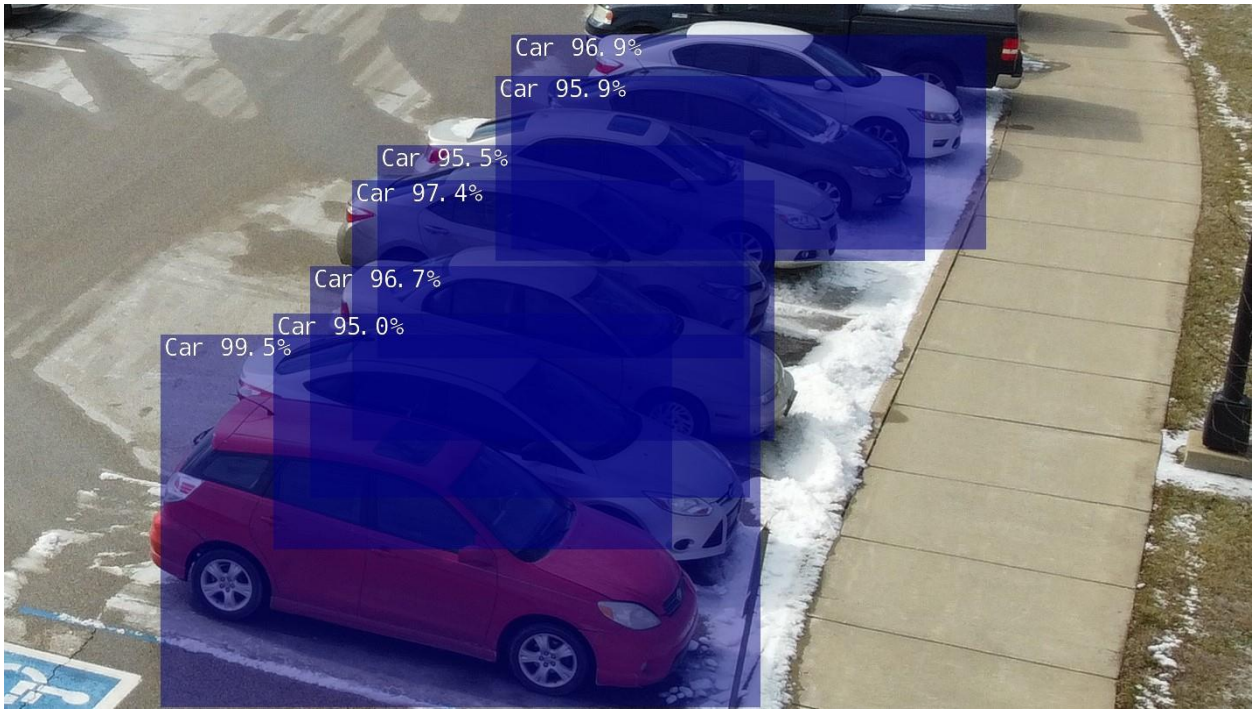


Figure 17. Vehicle Detection with Overlapping Cars

As the accurate count of vehicles present is critical for a purely image-processing based monitoring solution, obstruction of vehicles from the view of the camera was determined to be a high-risk failure mode of this design. In Figure 17, multiple vehicles are detected multiple times, while other vehicles are not identified. In this particular case, the number of detections matches the number of vehicles, but this is not due to proper detections. The faulty detections were not able to be eliminated through manipulating the detection threshold as the duplicate detections have high confidences reported. Due to the vehicles overlapping, compensating for incorrect detections through data post-processing is difficult as the detection bounding boxes overlap. Potential camera placements to avoid this failure mode are covered later in this report.

3.1.3 *Light Post Camera Placement*

An alternative camera placement to interior camera mounting is to mount cameras to each of the light posts in the lot. Figure 18 shows the light post locations and a potential coverage map for cameras mounted to light posts in lot J.



Figure 18. Camera Coverage Map with Light Post Locations

The proposed camera placements in Figure 18 would require multiple cameras on each post to achieve proper views of the proposed parking spaces. The parking spaces chosen in this figure limit the potential for obscured vehicles due to camera perspective. These positions were also chosen to limit image size to account for image resolution as is discussed below.

3.1.4 Infrastructure Limitations

Exterior mounted cameras introduce several design considerations. These design considerations include electric supply, weatherproofing, and data communication. The first issue is mounting cameras to the light posts would require single phase 120 V electric service to be run to the posts. These light posts are currently supplied with 208 V three phase power according to the USI Facility Operations and Planning worker Kendal Roeder. This issue will be discussed further in the cost analysis section of this report.

The second issue is exterior mounting introduces all electronics involved to water intrusion. The method in which this is addressed depends on the implementation. As the design of such an enclosure is outside the scope of this project, such a design will not be supplied. The following should be considered when making such an enclosure. Any component exposed to the

elements should meet the International Electrotechnical Commission ingress protection rating for the application following the standard IEC 60529. As the camera and/or enclosure will be a permanent installation, a solid foreign object rating of 6 should be used to prevent dust or dirt accumulation on internal circuitry. As the light posts will provide little to no protection from rainfall, a water ingress rating of at least 5 should be used to prevent water intrusion from heavy rainfall. In conclusion, all external components and enclosures should meet or be designed to meet a minimum rating of IP-65 [21].

The third issue with exterior mounting is data, be it video feeds or vehicle counts, will need to be transmitted to a central processing center. Figure 19 below shows the distance from the closest point on the B.E. Center to the furthest light post is approximately 365 feet (111 m).



Figure 19. Furthest light post from Business and Engineering Center. Adapted from [1]

As including a physical data connection to each light post is unnecessary for a single device connection, wireless communication would be needed to transmit data from the cameras to the central processing device. Two methods could be used for this communication. The first option would be to use a device to wirelessly transmit video data to a central processing computer such as the Jetson Nano used in this project. The second option would be to install a

Jetson Nano to each light post to process the image data. The vehicle/space count would then be transmitted to a central device to monitor the vehicle detections from all cameras and control the output to drivers. In this option, the enclosure would need to be designed with consideration for heat dissipation from the Jetson Nano. Both options will be considered later in the cost analysis section of the report.

Regardless of which data is transferred, wireless communication is necessary for this setup. To avoid FCC frequency restrictions, Wi-Fi communication offers the greatest communication range without sacrificing data transfer rate. To ensure devices do not interfere with other communications, any devices used should follow current IEEE 802.11 specifications. With a properly mounted exterior antenna, the furthest device, located approximately 111 meters from the B.E. Center, is well within the maximum reliable outdoor Wi-Fi range of approximately 300 meters [22].

3.1.5 Image Resolution Considerations

Further issues with some camera placements arise when image resolution is considered. Due to the way the SSD detection framework functions, all images are reduced to a resolution of 300×300 pixels. Once the image is in this resolution format, the detection network can only accurately detect objects with bounding boxes of at least 50×50 pixels (Hai et al.). Figure 20 and Figure 21 below show how the reduction in image resolution affects the smallest detectable object. The left side of each figure shows the original image, while the right side of each figure shows the reduced resolution image. The yellow hatched square overlaid on the compressed images illustrates the 50×50-pixel minimum effective bounding box.

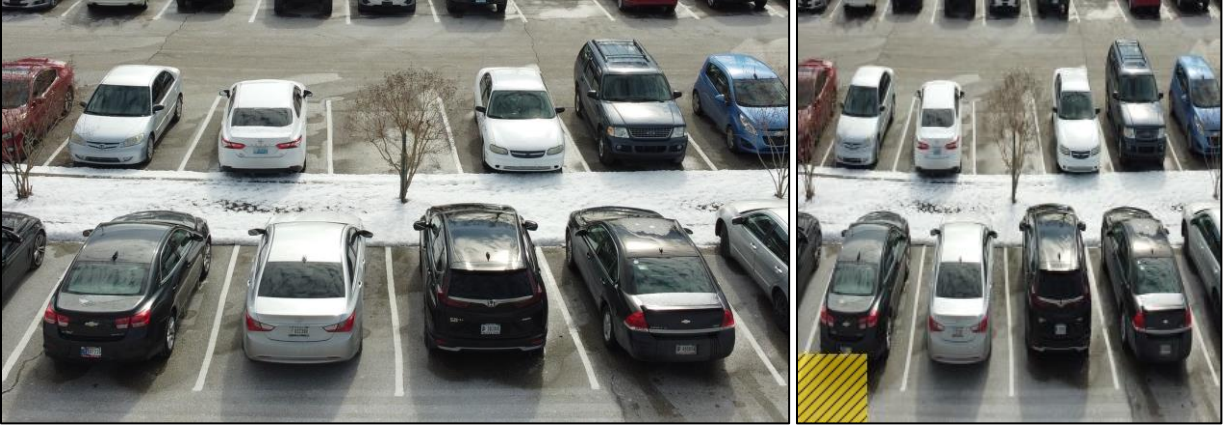


Figure 20. Effects of Image Compression on Smaller Detection Area



Figure 21. Effects of Image Compression on Smaller Detection Area

The two figures above illustrate an additional failure mode to be considered during camera placement. Any object in the image that is smaller than the 50×50-pixel region highlighted has a higher potential to be missed during image processing. Wider FOV cameras are more susceptible to this due to the higher horizontal compression to bring the image to a square aspect ratio.

In practice, the SSD Mobilenet can detect objects in bounding boxes smaller than 50×50 pixels, however, the results in such cases are unreliable, resulting in missed vehicle detections. Figure 22 below showcases how an increased view area results in missed detections. The colored boxes overlaid on the image represent the bounding box of an object. Blue boxes indicate the detection of a car, grey boxes indicate the detection of a truck, and teal boxes indicate the detection of a person.



Figure 22. Effect of Detection Area on Detection Results

Several failure modes of this detection model are seen in Figure 22. The first failure mode is missed detections of smaller objects. This was anticipated due to the restrictions caused by the image compression involved in the image processing. The confidence in the detections increases as the vehicles become relatively larger. At larger sizes, duplicate detections begin to appear, as can be seen with the truck in the 1.5 \times zoom image and four of the vehicles in the 2.5 \times zoom image. The confidence values on these detections are lower as the redundant detections are from training the model on overlapping or obscured cars. Further details regarding these errors will be discussed in the transfer learning section below. These duplicate detections can be filtered out by increasing the confidence threshold for positive detections or through data postprocessing.

3.2 DETECTION RESULTS AND TRANSFER LEARNING

3.2.1 Initial Results

As was previously mentioned, SSD Mobilenet is pre-trained on 91 classes of objects. Though several of these classes are various vehicle types, initial results revealed errors in the base model. Figure 23 below showcases some of the initial results tested on images from lot J. The left image shows how the initial model failed to detect multiple vehicles, and of the vehicles detected, three cars were classified as “bus”. The image on the right shows how, when supplied a non-ideal perspective, SSD-Mobilenet will try to classify the image with all 91 classes. In this case, the model has classified the entirety of the image as “suitcase”.



Figure 23. Example Missed and Incorrect Detections with SSD-Mobilenet

The errors in Figure 23 posed three main concerns. The primary concern were the missed vehicles detections in the left image. If this model was to be used for the purposes of this project, three filled spaces would be identified as empty despite the presence of vehicles. The misclassification of the three cars as “bus” was not considered a concern as the bounding boxes accurately identified the locations of the vehicles. As the final product does not need to identify the type of vehicle, misclassified vehicle types were considered not a failure mode. The misclassification found in the right image in Figure 23 was considered a potential failure mode. As multiple vehicles are grouped together and classified as a non-vehicle, this would result in an invalid vehicle count.

3.2.2 Model Improvement with Open Images Dataset

It was initially assumed that the detection issue was due to the number of detection classes present. To eliminate incorrect class labeling such as in the “suitcase” detection in Figure 23, a transfer learning approach was applied to the SSD Mobilenet architecture. Opensource

OpenCV image datasets from Open Images V7 were used to complete this initial retraining. Datasets for the classes bicycle, bus, car, person, truck, and van were used. As such, the retrained model was only able to classify detections within these six object classes. Figure 24 below shows the detection results with the retrained model.

The model was retrained over 30 Epochs.



Figure 24. SSD-Mobilenet Detection Results – Retrained with Open Images Datasets

The retrained model returned identical results for the first test image. The second image did not result in misclassification, but no cars were detected. As nothing was detected, it is unclear if the result changed because the suitcase class was removed or if the transfer learning model had improved. As the confidence percentages in the first image were unchanged, it was assumed that the model had not improved. This result indicated the retrained model only effectively reduced the number of detection classes but did not change the deep neural network.

The results in Figure 24 made it evident that multiple sources of error were causing the detection errors. To investigate the detection errors further, a live camera feed was used with a scaled model car. The camera placement relative to the car was varied to check the effect of perspective and relative vehicle size on detection results. Various portions of the car were also obscured to analyze the effect on the results.

From the live-feed testing, two main errors sources were identified. The first source of error was relative vehicle size within the image. When the relative vehicle size was less than approximately 25 percent of the total image, detections became unreliable. The second source of error was due to camera perspective. When the car was viewed from directly in front of or behind, or when viewed from above, the detection model consistently failed to detect the car.

Following the live-feed camera test, the Open Images datasets were reviewed for image content. Reviewing the dataset found that images within the set focused primarily on vehicle views from ground level with the side of the car visible. Most clear images within the dataset contain partial images of vehicles. Figure 25 below shows two example images from the “car” Open Images Dataset.

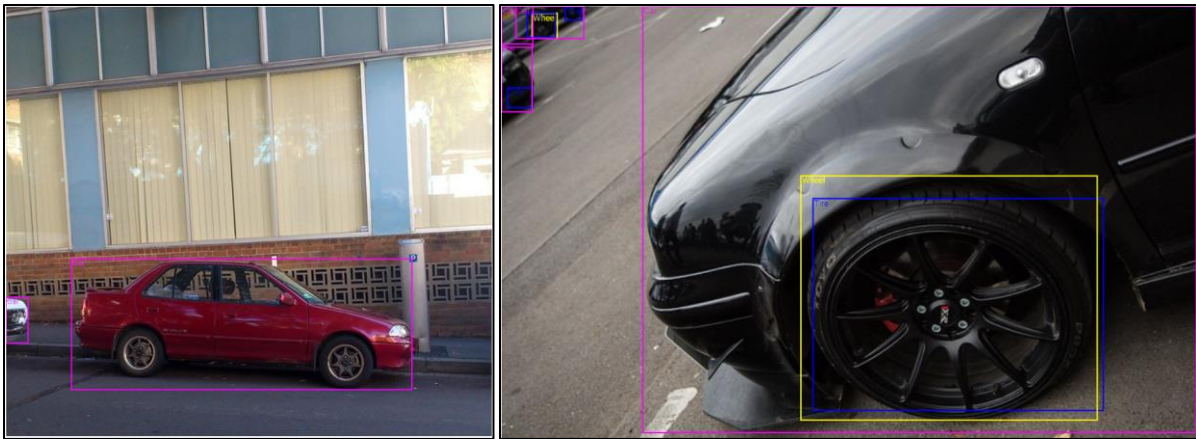


Figure 25. Typical Views of Car in Open Images Dataset. [23]

From the live-feed test and dataset review, it was determined that the model was not properly trained to detect vehicles from the aerial views necessary for this project. To improve results, the retrained model was retrained again utilizing a custom dataset.

3.2.3 Model Improvement with Custom Dataset

Aerial perspective images were captured from the second and third floors of the B.E. Center southwest stairwell. These images were cropped to include representations of the vehicle views anticipated in this application. The dataset included images of vehicles from multiple angles with varying amounts of vehicles obscured from view. The images were also cropped to include images with smaller relative vehicles sizes. These images were converted to a VOC image dataset using NI Label Studio.

The NI train_ssd.py program was used with this VOC dataset to retrain the SSD Mobilenet detection network.

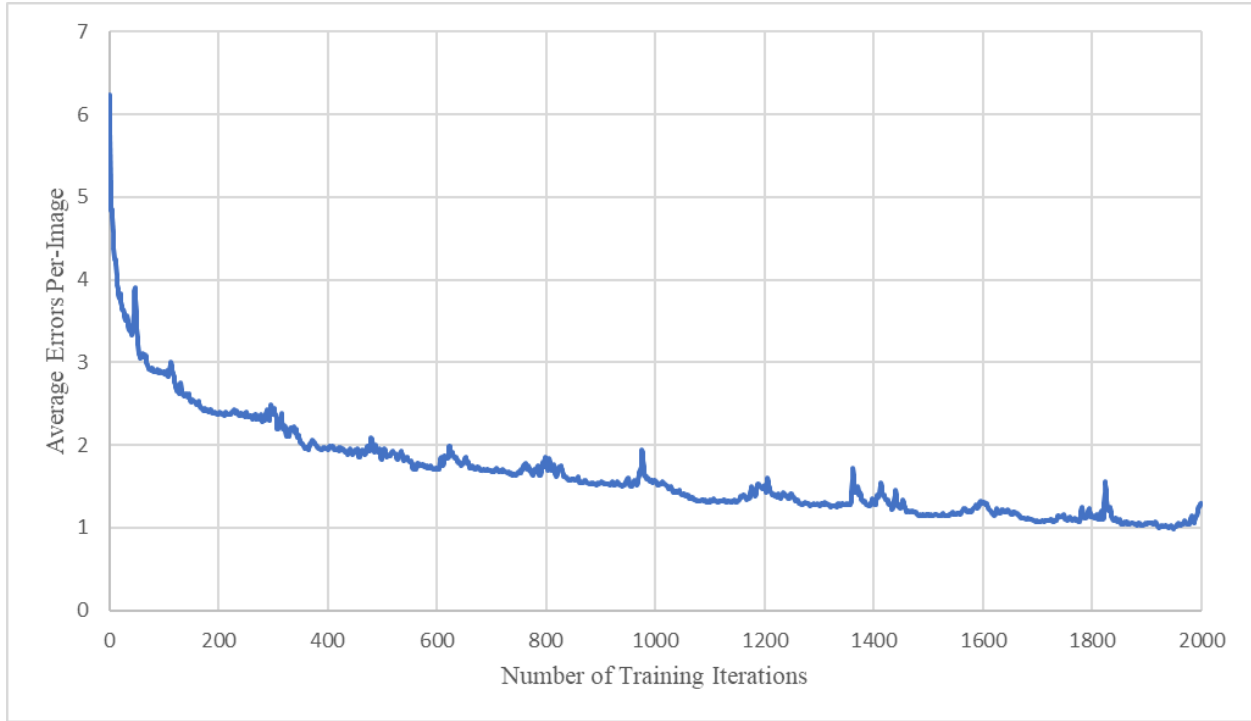


Figure 26. Detection Error Improvement with Additional Training Iterations

3.3 *COST ANALYSIS*

The cost for the suggested multi-camera implementation as well as traditional sensor-based solutions were evaluated. Due to the highly variable costs of components and labor, ranges of values were provided. Table 1 below shows a summary of the estimated costs of each system type. The following tables provide a more detailed cost breakdown of each system.

Table 1. Summary of System Costs

Space Sensors:	
Ultrasonic	\$ 71,750.00 - \$ 246,000.00
Induction	\$ 1,025,000.00 - \$ 2,050,000.00
Entrance Sensors:*	
Ultrasonic	\$ 1,400.00 - \$ 4,800.00
Induction	\$ 20,000.00 - \$ 40,000.00
Vision System:**	
	\$ 3,650.00 - \$ 9,650.00
Hybrid System:	
	\$ 3,270.00 - \$ 9,550.00

* Cost does not include calibration and maintenance costs

** Cost does not include labor costs

Table 2. Cost of Per-Space Sensor Implementation [24] [25]

	Sensor Type:	Cost Per Space	Total
Parking Space Sensors	Ultrasonic Sensors	\$ 350 - \$ 1,200	\$ 71,750 - \$ 246,000
	Induction Sensors	\$ 5,000 - \$ 10,000	\$ 1,025,000 - \$ 2,050,000

Table 3. Cost of Ingress-Egress Sensor Implementation [24] [25] [26]

	Sensor Type:	Cost Per Space	Total
Ingress/Egress Sensors	Ultrasonic Sensors	\$ 7 - \$ 23	\$ 1,400 - \$ 4,800
	Induction Sensors	\$ 98 - \$ 195	\$ 20,000 - \$ 40,000

Table 4. Cost of Multi-Camera Detection Implementation [15]

	Component Cost	Quantity	Total
Camera	\$ 100 - \$ 200	25 - 30	\$ 2,500 - \$ 6,000
Jetson Nano	\$ 150 -		\$ 150 - \$ 150
Data Transfer	\$ - - \$ 1,500		\$ - - \$ 1,500
Display	\$ 1,000 - \$ 2,000		\$ 1,000 - \$ 2,000
		Grand Total:	\$ 3,650 - \$ 9,650

Table 5. Cost of Hybrid Ingress-Egress Ultrasonic and Detection System [15] [24]

	Cost		
Ingress/Egress Sensors	\$ 2,100	-	\$ 7,200
Cameras	\$ 20	-	\$ 200
Jetson Nano	\$ 150		
Display	\$ 1,000	-	\$ 2,000
Grand Total:	\$ 3,270	-	\$ 9,550

The high number of sensors required to use a per-space sensor implementation results in a high cost for such a system. Due to the high cost associated with induction sensors as well as the construction required to install such a sensor, use of such sensors in general is not desirable for this application. Installation of an ultrasonic sensor ingress/egress sensor setup offers the lowest cost solution. However, this style of system does not account for vehicles in handicapped-accessible spaces. To monitor such spaces, additional per-space sensors would be needed, adding approximately \$6000-\$20,000 to the cost of such a system. This cost does not include labor costs associated with maintaining calibration of such a system. As this type of system is prone to losing count over time from erroneous detections or one direction detections such as grounds equipment entering through the entrance and leaving the lot elsewhere, ingress/egress type systems require regular count checks and calibration to maintain an accurate count [27].

The cost breakdown for a camera-based detection system can be seen in Table 4. The costs associated with the number of cameras necessary for the solution developed in this report exceeds the amount that could be implemented by the department. The cost estimated above does not include the labor costs associated with installation of the cameras or the cost associated with the potentially necessary electrical infrastructure changes. Further investigation into these labor and infrastructure costs would be necessary before further pursuing such a solution.

An option that could offer a lower installation cost would be to implement a hybrid system can be seen in Table 5. An ultrasonic sensor ingress/egress system could be paired with a detection-based system. The ingress/egress sensors would provide a total vehicle count while a camera mounted on the third floor would be capable of monitoring the handicapped-accessible spaces in lot J. This option would minimize the required infrastructure changes associated with the installation of such a system.

4 CONCLUSIONS AND RECOMMENDATIONS

4.1 PROJECT CONCLUSION

Parking lot J on the USI campus provides the primary parking for the Business and Engineering Center and Liberal Arts Building. During peak class times, the 207 parking spaces fill quickly, leading to wasted time and frustration for drivers looking for a spot in the lot. To provide drivers with a notification if parking USI parking lot J was full, it was originally suggested to mount a camera on the third floor of the B.E. building in conjunction with a vision learning setup. To investigate this, a vision learning setup was implemented using an Nvidia Jetson Nano and associated Nvidia programs and libraries. The vision learning system was based on an object detection architecture using SSD Mobilenet, though the detection architecture had to be retrained to accurately detect vehicles. Testing using this setup unveiled several issues preventing such a solution from working as originally suggested.

- The image compression associated with object detection leads to cars being difficult to detect from the perspective
- The perspective from the third floor leads to some spaces not being visible

Improving the issues associated with image compression could be solved by switching to a different detection architecture such as YOLO, but this would require methods differing from those captured in this report. The issues associated with perspective could be solved by implementing a system of cameras mounted to the light posts in lot J, though the added complexity and costs associated with such a system remove advantages of the proposed solution over existing sensor based or commercially available solutions. A single camera setup in conjunction with ultrasonic sensors at the entrance of lot J could potentially provide a more cost-effective solution to the problem. Such a solution could provide an overall car count through sensors at the lot entrance with a detection system monitoring the handicapped-accessible spaces.

4.2 FUTURE RECOMMENDATIONS

Further investigation into implementing ultrasonic sensors at the entrance of parking lot J with a detection-based system monitoring handicapped-accessible spaces should be conducted to evaluate the project feasibility.

REFERENCES

- [1] Google, "Google Maps," [Online]. Available: maps.google.com. [Accessed 2023].
- [2] J. Chinrungrueng, S. Dumnin and R. Pongthornseri, "iParking: a Parking Management Framework," *International Conference on ITS Telecommunications*, 2011.
- [3] O. Dokur, "Embedded System Design of a Real-time Parking Guidance System," 2015.
- [4] T. N. Hilleary, *A Radar Vehicle Detection System for Four-Quadrant Gate Warning Systems and Blocked*, Federal Railroad Administration, 2012.
- [5] M. A. Abidin and R. Pulungan, "A Systematic Review of Machine-vision-based Smart," *Scientific Journal of Informatics*, pp. 213-227, 2020.
- [6] Chetu, "Parking Management Software Solutions," [Online]. [Accessed January 2023].
- [7] intuVision, "intuVision Parking," [Online]. [Accessed January 2023].
- [8] A. O. Kotb, "Smart Parking: Guidance, Monitoring and Reservations," 2016.
- [9] Signal-Tech, "Parking Space Available and End of Aisle Signs with LED Counters," 2023. [Online]. Available: https://www.signal-tech.com/products/parking/space_available_and_end_of_aisle.
- [10] S. Sarkakr, M. W. Totaro and K. Elgazzar, "Intelligent Dron-based Surveillance: Application to Parking Lot Monitoring and Detection," *Unmanned Systems Technology*, 2019.
- [11] C. Jang and M. Sunwoo, "Semantic segmentation-based parking space detection with standalone around view monitoring system," *Machine Vision and Applications*, pp. 309-319, 2019.

- [12] J.-Y. Chen and C.-M. Hsu, "Around View Monitoring-Based Vacant Parking Space Detection and Analysis," *Applied Sciences*, 2019.
- [13] M. Phadtare, V. Choudhari, R. Pedram and S. Vartak, "Comparison between YOLO and SSD Mobile Net for Object Detection in a Surveillance Drone," *International Journal of Scientific Research in Engineering and Management*, 2021.
- [14] P. Ramaswamy, *IOT smart parking system for reducing Green House Gas Emission*, 2016.
- [15] NVIDIA, "Getting Started with Jetson Nano Developer Kit," [Online]. [Accessed February 2023].
- [16] D. Franklin, "Jetson Inference - Object Detection," 18 June 2019. [Online]. [Accessed February 2023].
- [17] NVIDIA, "CUDA Toolkit 12.1 Downloads," 2023. [Online]. [Accessed February 2023].
- [18] NVIDIA, "NVIDIA cuDNN," 2023. [Online].
- [19] J. Brownlee, "A Gentle Introduction to Transfer Learning for Deep Learning," 20 December 2017. [Online]. [Accessed February 2023].
- [20] I.-G. Lee, D. B. Kim, J. Choi, H. Park, S.-k. Lee, J. Cho and H. Yu, "Wi-Fi HaLow for Long-Range and Low-Power Internet of Things: System on Chip Development and Performance Evaluation," *IEEE Communications Magazine*, vol. 59, no. 7, pp. 101-107, 2021.
- [21] International Electrotechnical Commission, *IEC 60529*, 2.0 ed., 2019.
- [22] S. Kaushik, "An overview of Technical aspect for WiFi Networks Technology," *International Journal of Electronics and Computer Science Engineering*, vol. 1, no. 1, pp. 28-34, 2012.

- [23] Google, "Open Images Dataset V7," 2023. [Online]. Available:
<https://storage.googleapis.com/openimages/web/visualizer/index.html>.
- [24] United States Department of Transportation, "Advanced parking management systems cost," 21 4 2008. [Online]. Available:
<https://www.itskrs.its.dot.gov/its/benecost.nsf/ID/ee8ccecead836eb6852573e2006bc427>.
- [25] B. Lozano, "Space Available for Parking System's Growth," 31 3 2021. [Online]. Available:
<https://www.facilitiesnet.com/facilitiesmanagement/article/Space-Available-for-Parking-System8217s-Growth--19195#:~:text=Facility%20Parking%20Count%20System&text=The%20cost%20for%20these%20types,basic%20lot%20open%2Ffull%20sign..>
- [26] J. Lamontagne, "What is the Return on Investment for a Smart Parking System?," 27 9 2021. [Online]. Available: <https://www.dimonoff.com/news/what-is-the-roi-for-a-smart-parking-system/>.
- [27] C. Sobie, "Life Cycle Cost Analysis of Vehicle Detection," Western ITE, 2016.

APPENDIX

APPENDIX A:



APPENDIX B: DETECTION PYTHON SCRIPT

```
import jetson.inference
import jetson.utils
import math

# Define detection model and threshold
net = jetson.inference.detectNet("ssd-mobilenet.onnx", threshold=0.8)
# Define camera parameters
camera = jetson.utils.gstCamera(1280, 720, "/dev/video0")
# Setup display to show detections (not needed for space count)
display = jetson.utils.glDisplay()

# Define classes to look for and number of spaces
classes = ["Bus", "Car", "Motorcycle", "Truck", "Van"]
spaces = 6
hspaces = 0

# Open bounding box file and extract data
boundsx = []
boundsy = []
with open("boundsfile.txt", "rt") as boundsfile:
    i0 = 0
    for line in boundsfile:
        if ((i0 % 2) == 0):
            boundsx.append(line)
        else:
            boundsy.append(line)

# Open handicapped spaces location file and extract data
hspacex = []
hspacey = []
with open("hspacefile.txt", "rt") as hspacefile:
    ih = 0
    for line in hspacefile:
        if ((ih % 2) == 0):
            hspacex.append(line)
        else:
            hspacey.append(line)

# Generate equations of lines for bounding box
imb = 0
for x in boundsx:
    m[imb] = (boundsy[imb+1] - boundsy[imb]) / (boundsx[imb+1] - boundsx[imb])
    b[imb] = boundsy[imb] - m[imb] * boundsx[imb]

    imb = imb+1

# Run detection network
# in this case, it opens a window and shows detections
# this while case can be changed if visual output is not wanted
while display.IsOpen():
    img, width, height = camera.CaptureRGBA()
    detections = net.Detect(img, width, height)
    display.RenderOnce(img, width, height)
    display.SetTitle("Object Detection | Network {:.0f} FPS".format(net.GetNetworkFPS()))
```



```

objects = detections
i1 = 0
for x in objects:      # pull all detection locations to a variable
    types.append(detections[i1].ClassID)
    i1 = i1+1

i2 = 0
found = []
foundL = []
foundR = []
foundT = []
foundB = []
for x in classes:     # extract only wanted detection classes
    if (types[i2] == 1):
        found.append(detections[i2].Center)
        foundL.append(detections[i2].Left)
        foundR.append(detections[i2].Right)
        foundT.append(detections[i2].Top)
        foundB.append(detections[i2].Bottom)
    i2=i2+1

i3 = 0
count = 0
for x in found:       # this loop checks for number of vehicles in bounding area
    if (found[i3,1] > ((found[i3,2] - b[0])/m[0])):
        if (found[i3,1] < ((found[i3,2] - b[2])/m[2])):
            if(found[i3,2] < (m[2]*found[i3,1] + b[2])):
                if(found[i3,2] > (m[4]*found[i3,1] + b[4])):
                    count=count+1
    i3=i3+1

i4 = 0
i5 = 0
hcount = 0
for x in hspacex:    # this loop checks for number of vehicles in handicapped spaces
    for y in found:
        if (foundL[i4] < hspacex(i5)):
            if (foundR[i4,1] > hspacex(i5)):
                if(foundT[i4,2] > hspacey(i5)):
                    if(foundB[i4,2] < hspacey(i5)):
                        hcount=hcount+1
    i4=i4+1
    i5=i5+1

openspaces = spaces-(count-hcount)
hcopenspaces = hspaces-hcount
print('There are ' + openspaces + ' open parking spaces)
print('There are ' + hcopenspaces + ' open handicapped accessible parking spaces)

```

Appendix C: ABET Outcome 2, Design Factor Considerations

Table C.1, Design Factors Considered

Design Factor	Page number, or reason not applicable
Public health, safety, and welfare	While reducing unnecessary traffic in an area of high pedestrian traffic does impact public safety, an analysis of the degree of this impact was considered out of the scope of this project.
Global	While parking lot management is a global problem, this project was focused specifically on parking lot-J of the University of Southern Indiana campus and, therefore, does not have a global impact.
Cultural	This project does not have a cultural impact.
Social	Though cameras in public places can offer a social dilemma, the cameras used in this project were not used to record or save footage and therefore do not have a social impact.
Environmental	See page 7
Economic	See page 31
Ethical & Professional	This project was found to not violate any section of the NSPE code of ethics.
Reference for Standards	[21] – referenced on page 23