Real-Time Parking Sign Detection for Smart Street Parking

Yin Jin

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Computer Science and Systems

University of Washington

2022

Reading Committee:

Juhua Hu, Chair

Wei Cheng

Program Authorized to Offer Degree:

Computer Science and Systems

University of Washington

Abstract

Real-Time Parking Sign Detection for Smart Street Parking

Yin Jin

Chair of the Supervisory Committee:
Juhua Hu, Chair
Department of Computer Science and Systems

Correctly interpreting the complex parking sign and finding a suitable parking spot is always a challenging task to do in a short time. An automatic parking sign understanding assistance can improve the efficiency of a driver's daily life. Besides, as Tesla becomes popular, automatic driving is also becoming a prevalent task. However, the current studies on self-driving steering command mainly focus on the driving part. The smart parking task is still an open task. In this thesis, we aim to handle a subtest of the automatic parking sign understanding problem, that is, real-time parking sign detection, to provide real-time parking sign localization for further sign interpretation. Specifically, with a real-time video streaming of the street view as input, we aim to detect parking signs on the street that will inform the parking rules over the current road. In this thesis, we achieved two main goals: 1) generating a diverse parking sign dataset with a size over 4,000 that covers complex street view; 2) training a well-performance real-time parking sign detection model with Yolov5. Our parking sign detection model achieves a mean average precision at Intersection over Union threshold 50% (mAP@.5) of 0.968 and an inference speed of 6.1ms per image that meets the real-time detection requirement. Moreover, the model size is only 14.4 MB which is small enough to fit in small and less powerful devices like mobile phones or autonomous cars.

Table of Contents

List of Figures

List of Tables

1. Introduction

Many drivers have the experience that they are going to be late for something. Although they have already arrived at the location, they cannot find a parking spot. Even when the driver is not catching the time, it is still annoying when the driver notices there is a parking spot right after the driver passes it, or right after the driver parked, the driver notices that he cannot park there. Therefore, an assist that can detect parking signs in real-time and provide parking instruction is helpful, especially for a new driver who lacks the driving and parking experience. An automated real-time parking sign detection can improve the efficiency of a driver's daily life.

In addition to the need for drivers, real-time detection has a tremendous future developing prospect for autonomous cars. As Tesla becomes popular, automatic driving is also becoming a prevalent task. The current studies on the self-driving steering command mainly focus on the driving part. A well-designed self-driving system works on different road conditions like local roads, highways, or unpaved roads [12]. However, the specific task of finding the parking spot is still open.

Therefore, this thesis aims to address the Real-Time Smart Parking task, a subtask of automatic driving. With the real-time video stream from a front-facing camera as input, this thesis trains a model to detect parking signs on the street that inform the parking rules of the current road. The detection result can provide instructions to a human driver or an autonomous car in the future. To solve the real-time detection task, the You Only Look Once (Yolo) algorithm is a potential solution [3]. Yolo has a unified detection framework that combines the bounding box detection and a classifier into a single neural network to help detect an object in one image. The Yolo can be fast enough to achieve real-time detection using this unified detection design. However, Yolo has a drawback on accuracy compared to RCNN [2], a powerful two-stage algorithm. Fortunately Yolo v5, using Mosaic Data Argument, solves the accuracy problem while keeping the model size small

[11]. However, Yolo v5 has not been explicitly used for the parking sign detection application in the wild and can be very complicated on the street.

Therefore, we propose to build a real-time parking sign detection model based on Yolov5 with our customer parking sign dataset and evaluate its performance on both aspects of images and real-time video streams. We overcome three main challenges for this task: real-time detection speed, small model size, and customer parking sign dataset. Our current model can detect 163 parking sign images per second with the guaranty of detection accuracy. Specifically, the mean average precision at Intersection over Union threshold 50% (mAP@.5) is 0.968 on the testing dataset. The high detection speed and accuracy allow our efficient parking sign detection model to detect parking signs when driving around. In addition, our model is only 14.4MB, which is small enough to embed on a small device like a mobile or an autonomous car. We also generated a 4,191 sized dataset that half of them are parking sign images and half of them are street view video frames.

This thesis is organized as follows. In Section 2, we first introduce some related work in the literature. Section 3 describe the details of Yolo v5 to solve our real-time parking sign detection problem. Section 4 reports the experiments conducted, which is followed by a conclusion with future direction discussion in Section 5.

2. Related Work

Our task is under the scope of object detection, which is a combination of object localization and object classification. Currently, there are two types of solutions: two-stage detection and Unified Detection. Most early object detection approaches like the Deformable Parts Model (DPM) [1] and Region-Based Convolutional Network method (R-CNN) [2] are two-stage detection. They split the object detection task into two components: an object bounding box proposal and a

classifier. This design is hard to optimize. The Unified Detection Design combines object proposal and objects classifier into a single regression problem, which is easy to optimize.

2.1. Two Stage Detection

2.1.1. Deformable Parts Model (DPM)

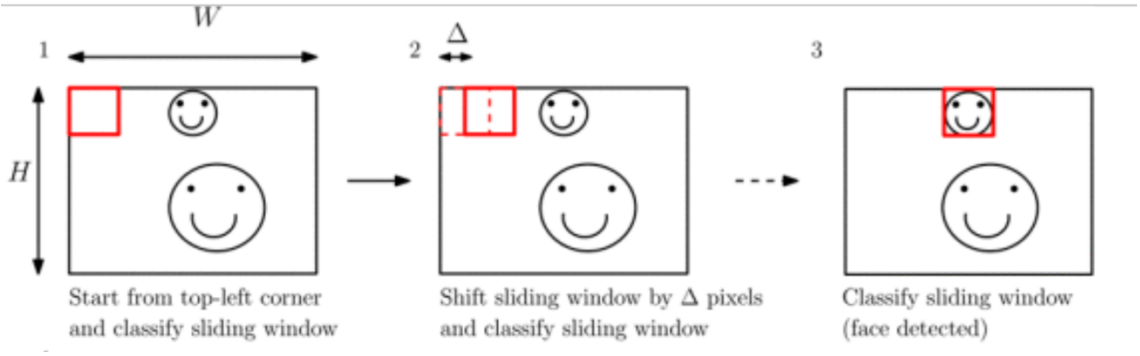

Figure 1:Sliding Window Approach [21]

Deformable Parts Model (DPM) [1], issued in 2010, uses a straightforward way to propose bounding boxes: the sliding window approach. As shown in Figure 1, this approach starts at the top left corner of the input image with a fixed-size window. Then it slides the window from left to right and up to down, with a fixed step. Eventually, the window will stop at the bottom left corner. DPM then applies a classifier on all proposed windows. However, this straightforward bounding box way requires massive computation to guarantee accuracy. First, the step between windows has to be small enough to make sure not to miss any object. However, the small step will lead to a substantial redundant computation. In addition, a fixed bounding box will also limit the performance of the model since the object size in the image is unlikely to be the same. A solution to this problem will be to use multi-windows size.

2.1.2. Region-Based Convolutional Network method (R-CNN)

Like DPM, the Region-Based Convolutional Network method (R-CNN) [2], published in 2014, is a two-stage detection method. They apply the object classification model of these proposed bounding boxes and keep the one above a threshold. They then use post-processing to eliminate the duplicate objects, increase the bounding box accuracy, and recalculate the confidence of bounding boxes. Instead of the sliding windows approach, RCNN uses a region proposal. In the early version of RCNN, they randomly generated a 2k bounding box random size and random location. The region proposal approach reduces a lot of redundant computation compared to DPM and provides high accuracy. Later, the Fast R-CNN, a faster version of the R-CNN, is designed with a more efficient bounding box and classifier. Compared to the early study, the Fast R-CNN's performance on the PASCAL VOC 2012 dataset is nine times faster on training and 213 times faster on testing [5]. However, it still needs to take around 40 seconds per image at test time, far from real-time detection.

2.2. Unified Detection Design

In 2015, the You Only Look Once (Yolo) [3] algorithm was proposed. Yolo took the lesson from the traditional method. Instead of splitting the task into several parts, Yolo introduced a unified detection design that uses a signal neural network to predict the bounding box and find the probability of the classes simultaneously. The signal network provides a chance to optimize the whole process through the pipeline. Yolo was the fastest algorithm at that time. The standard version can achieve 45 FPS, and the faster version can reach 150 FPS. The early edition report claims that Yolo has a drawback on accuracy compared to the state-of-art detection system.

The 4th version of Yolo [4], published in April 2020, solved the accuracy problem. Yolo v4 uses Mosaic data augmentation to extract the features, which is the crucial point of the Yolo v4 breakthrough on accuracy. Several months later, Glenn Jocher, acknowledged as the creator of mosaic augmentation, released yolov5 [11]. Both yolov4 and yolov5 use Mosaic data augmentation and achieve breakthrough accuracy problems. Yolo v5 is beating any other choice on its accessible features, for example, AutoAnchor. When we input customer data, all YOLO anchor boxes are auto-learned in YOLOv5. It can cut off many manual workloads and prevent careless mistakes. This accessible feature allows developers with customer datasets to focus on the object detection task. Thereby, Yolo v5 is more efficient. Another astrictive feature of the Yolo v5 suite is the model's size, since we require the model to be embedded on mobile devices. The post result of the Yolo v5 shows its weight file is nearly 90% smaller than the weight file of Yolo v4 [11]. However, Yolo v5 has not been explicitly used for the parking sign detection application in the wild and can be very complicated on the street. Therefore, in this thesis, we built a real-time parking sign detection model based on Yolo v5 with a self-generated dataset and evaluated its performance on both aspects of images and real-time video streams.

2.3. Parking Sign Detection

2.3.1. Smart Street Parking

A similar take was down by Jiang for Smart Street Parking in 2019 [20, 22]. They collected a parking sign dataset and trained a detection model with RetinaNet based on it. They did a great job on the accuracy, which claims to have mAP@.5 of 0.984. However, according to their report, they can only detect 17 frames per second, which did not reach the real-time requirement.

2.3.2. Smart Cities

Another relative work is Smart Cities [24], done by Faraji et al. They developed a slightly different technique for reading street parking signs. The authors unified the two phases of sign detection and interpretation into one step by assuming that each sign carries one parking regulation. They assume there are only three types of signs: no-stopping, no-parking, and parking-allowed. However, the parking rule is far more complex in the real world. For example, a handicap sign is a restricted access for a small group of people instead of a general user. Besides, how long the driver can park there and whether the driver needs to pay are essential parking rules that their model misses.

3. Methodology

Considering that Yolo v5 is both efficient and effective on other real-time object detection tasks [16, 17, 18], we propose to use it solve our real-time parking sign detection task. Concretely, using a Unified Detection Design, Yolo v5 can detect 140 images per second on other tasks reaching the real-time requirement [9]. Besides, Yolo v5 uses Mosaic Data Augmentation to improve the accuracy. We describe the details as follows.

3.1. Yolo v5: Unified Detection Design

The unified detection design that Yolo used is combining bounding box proposal and object binary classifier into a single network. As summarized in Figure 2, the algorithm will first split the image into $S \times S$ grides. It can predict at most B parking signs' bounding boxes whose center falls into that specific gride. Each bounding box will contain five predictions: x, y, w, h, c. (x, y) represents the location of the center of the parking sign object. The w is the width and the h is height. These two numbers represent the size of the parking sign object. The c represents the confidence score of the corresponding bounding box, which includes the

possibility that the bounding box contains a parking sign and the accuracy of that bounding box compared to the true one. This possibility is defined as

$$c = P(Obj) * IOU_{pred}^{truth}$$

$P(Obj)$ can be two values 0 or 1. For 0 it means there is no parking sign object existing in the cell, then confident score c is equal to 0. If P(Obj) =1, then confident score c is equal to IOU_{pred}^{truth}, which represents Intersection Over Union (IOU) between the predicted box and the ground truth. IOU use to evaluate the overlap degree between the predicted box and the ground truth. Instead of predicting each bounding box's classes as RCNN does, Yolo only calculates the conditional probability of all C classes for each grid, regardless of the number B, given the event of a grid cell containing an object. This conditional probability shows in the following formula:

$$\mathrm{C} = \mathrm{P}(\mathrm{Class_i}|\mathrm{Object})$$

In the inference stage, the parking sign class confident score for each bounding box, which is represented as $\mathrm{Pr}(\mathrm{Class_i}) * \mathrm{IOU}_{\mathrm{pred}}^{\mathrm{truth}}$, can be directly calculated according to confident score c for each bounding box and conditional parking sign class probability C.

$$\mathrm{Pr}(\mathrm{Class_i}|\mathrm{Object}) * \mathrm{Pr}(\mathrm{Object}) * \mathrm{IOU}_{\mathrm{pred}}^{\mathrm{truth}} = \mathrm{Pr}(\mathrm{Class_i}) * \mathrm{IOU}_{\mathrm{pred}}^{\mathrm{truth}}$$

Therefore, in our parking sign detection application, each image will split into $S\ X\ S$ grid cells. For each cell, the model can predict B bounding boxes with five predictions and $C(i.e.,1)$ class probability if there is a parking sign. Thereby, the model can be considered as $S \times S \times (B * 5 + C)$ tenser. For example, if $S = 7, B = 2, C = 1$. Then the final prediction should be a tensor with a size of $7 \times 7 \times 11$.

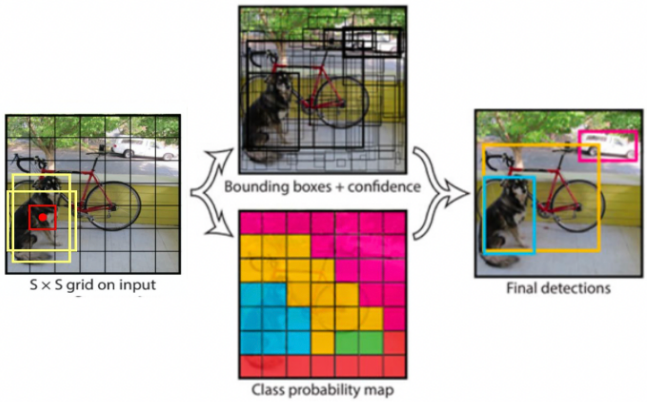

Figure 2: The Unified Detection Design [3]

3.2. Yolo v5: Activation function

Unlike Yolo v4, Yolo v5 used the Leaky ReLU activation function for the hidden layer and the Sigmoid activation function for the final detection layer [14]. The reason to chooses this set of activation functions instead of the Mish function that is uesd in Yolo v4, is that it is less expensive, even though Mish can improve the accuracy. This can help improve the efficiency as required by our task.

3.3. Yolo v5: Cost Function

The Yolo's cost function can be divided into three parts: bounding boxes regression score, existence of the object, and classifier score. Each bounding box contains five predictions: (x, y, w, h, c). The first three summand (i.e., Eqn. 1-3) evaluate all these five in groups of location(x, y), size (w, h), and confidence (c) of this bounding box. In Eqn. 4, this loss function also considers when there is no parking sign in an assigned position. Each grid cell can detect B objects, but usually, the number of the object is less than B. The last part (i.e., Eqn. 5) evaluates the accuracy of the classifier for each class. This loss function uses parameters λ_{coord} and λ_{noobj} to balance the weight between there is an object and there is no object.

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \quad (1)$$

$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \quad (2)$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 \quad (3)$$

$$+ \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2 \quad (4)$$

$$+ \sum_{i=0}^{S^2} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c) \right)^2 \quad (5)$$

3.4. Yolo v5: Data Augmentation

Figure 3: Mosaic Data Augmentation [4]

The input street-view images are loaded into a data loader during the input stage, containing data augments. A good data augment loader will transform the input images into training data, and more importantly, it explores the existing training data to fit a broader range of situations. Yolo v5 includes three main enhancements: Scaling, Color Space Adjustment, and Mosaic data augmentation. Since this thesis solves real-time detection problems with the input of a front

view camera's video, the parking sign's size is expected to be extremely small when far away and larger when it gets closer. Therefore, the scaling feature is suitable for real-time detection problems. The Mosaic augmentation was invented by Glenn Jocher [4] and was first introduced in Yolo v4. The Mosaic combines four existing training images into a new single image, as in Figure 3. Mosaic data augmentation will make the model perform better in sheltered and translation, alter the number of objects in sample images, and reduce batch-size demand. Besides, most images will only have one bounding box due to our training dataset's specialty. Mosaic will expand the number of the bounding boxes to a range between 0 to 4. Finally, most importantly, the reduction of batch-size demand means fewer requirements on the GPU memory. This advantage will lead to less computational cost on the training stage.

3.5. Training Cycle: Accumulate Dataset

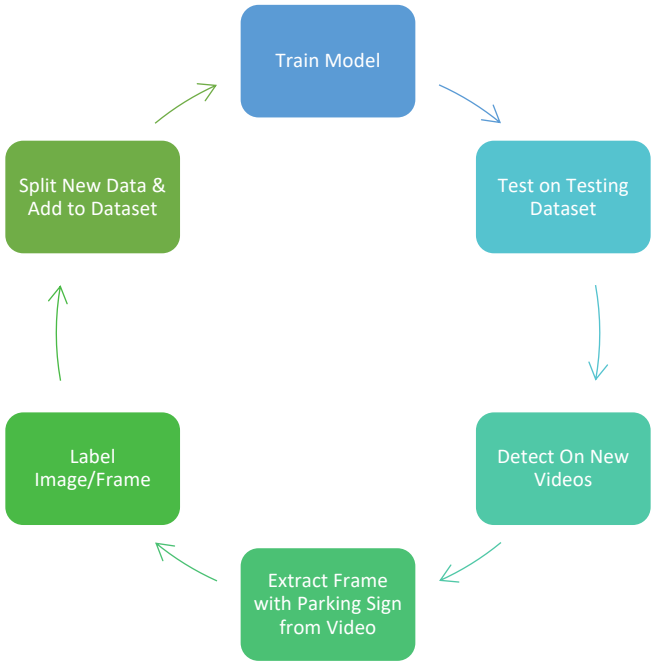

Figure 4: Training Cycle

To our best knowledge, the study on parking sign detection is very limited and the available parking sign data is also very limited and biased. For example, Irshad et al. [23] collected a

street parking sign data set from only San Francisco. Therefore, we collected a dataset containing 2,097 parking sign images covering ten states around the US. However, these data, which are collected through google street view or taken by a human being, is limited to real-time detection for the following reason. First, the angle of the google street view or photo taking is different from a dash cam's video frame (i.e., the scenario of our application). Second, parking signs in google street view images are too small to do detailed parking sign understanding. Finally, images taken by human beings are often very clear and directly facing the camera. In real life, most parking signs captured by a dash-camera are sheltered or have some angle variations. Therefore, we propose to start from our collected 2,097 images and use the training cycle in Figure 4 to accumulate a bigger and more diverse street view parking sign dataset. In this training cycle, we accumulated a more diverse dataset.

We first train a weak detector using the small parking sign image data collected, we will then test this model on a new video collected using the front dash-cam installed in the car. Each video contains a certain number of parking signs from a specific city in US. For those wrong detections, we then extract the frames as additional images and label them with Computer Vision Annotation Tool (CVAT) [19]. CVAT will export a new set of labeled data, which will be added to the training data to improve the model’s performance.

Eventually, we stop this training cycle when our model performs well on unseen videos. In this thesis, this training cycle adds 2,112 new street view frames. Therefore, we currently have a parking sign dataset with the size of 4,191.

3.6. Inference:

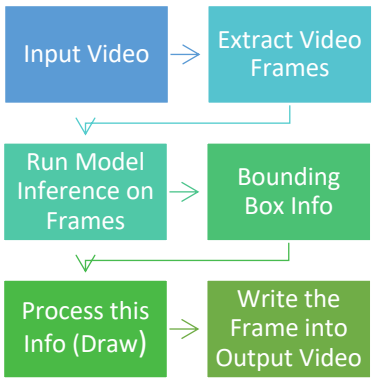

Figure 5: Inference on Video

With the real-time video stream from a front-facing camera as input, this thesis runs a pre-trained model to detect parking signs on the street that inform the parking rules of the current street. The first step of this inference is to extract video frames from the input video one by one. For each frame, we run the pre-trained model on it. The model will provide the bounding box information of parking signs in the current frame. Then we process this information and draw a bounding box around each parking sign object. After that, we can write each frame with bounding boxes into the output video. We then either show the video stream or save the video file.

4. Experiment

4.1. Set Up

4.1.1. Evaluation Metric: Average Precision (AP)

We evaluated the detection model on 3 aspects: accuracy, speed, and size of the model. For accuracy, we use Average Precision (AP), a popular evaluation metric for object detection. To find AP, we first need to find Precision, Recall, and Intersection over Union (IoU). For an object, there is a bounding box called ground truth. If there is no predicted box

overlapping with the ground truth box, then we call it False Negative (FN). If there is a predicted box, then we can find IoU by the formula:

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

For the calculated IoU greater than a defined threshold, we defined it as True Positive (TP) and False Positive (FP) otherwise.

Precision measure correctness in the formula (i.e., among all the prediction, how many are predicting correctly).

$$Precision = \frac{TP}{TP + FP}$$

Recall measures the ability to find all positive. (i.e., if there is an object, how many of them are predicted correctly).

$$Recall = \frac{TP}{TP + FN}$$

Then we can draw a Precision-Recall curve, let the x-axis as Recall and y-axis as Precision. The Average Precision is the area below this curve. The different thresholds will lead to different Precision and Recall, which leads to different Average Precision. Therefore, in some context, we have AP@.5, means AP with IoU > 0.5.

4.1.2. Environment Set UP

In this thesis, we train and test our model on the Google-Colab. As shown in Figure 6, the GPU computing processor we used is Tesla P100-PCIE-16GB with 16280MiB memory.

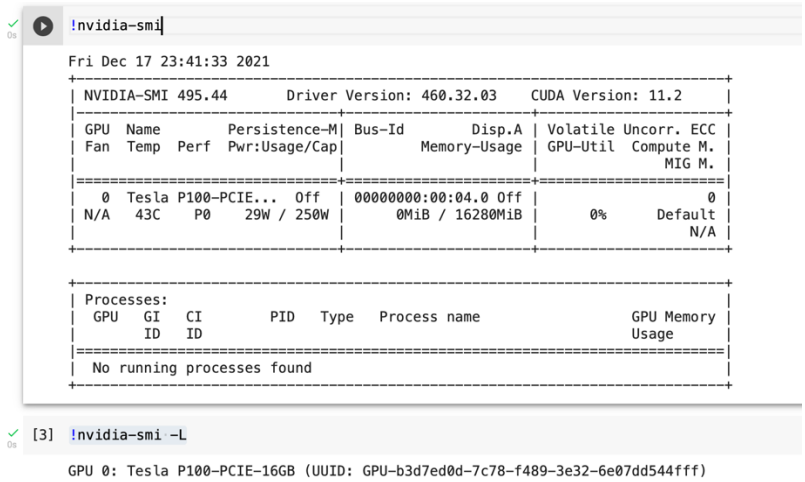

Figure 6: Our GPU Setting

4.1.3. Hyperparameters Setting

According to Yolo v5 requirements, we prepare a ymal file including train, validation, test folders path. The path to the test dataset is not required for the training stage. The ymal file also needs to include the number of classes (nc) and a list of strings for class names. For our project, we only have one class which is Parking Sign. The example of the yaml file is shown in Figure 7.

```
[ ] %%writefile parkingSignDec.yaml
    train: /content/ParkingSign_dataset/images/train/
    val: /content/ParkingSign_dataset/images/val/
    test: /content/ParkingSign_dataset/images/test/

    nc: 1
    names: ["Parking Sign"]
```

Figure 7: yaml File for Parking Sign Model

Besides the configuration of yaml file, we also need to set the appropriate hyperparameters for each training. We used 640 default "img" which represents image size, this number has to be a multiplication of 32 due to the Yolo v5 architecture. "batch" means batch size. Most of the time, the bigger the batch size, the higher accuracy. However, the batch size is limited by the memory. Also, if the batch size is too big, it may lead to the overfitting of the model. For this thesis, we used the biggest batch size capable of GPU memory. Then, we use the

learning curve from both training and validation data to check if overfitting or underfitting happens. Epochs represent the number of iterations of the training. We used 200 epochs, which allows the learning curve to converge. Hyperparameter data should follow the path to the config yaml file. Hyperparameter "weights" is the starting point of the training. We can pick from yolov5s.pt, yolov5m.pt, yolov5l.pt, yolov5x.pt. In the end, we set the hyperparameter "device" to cuda:0, which means that we want to use GPU to train the model. Besides, Yolo v5 provides two choices for the optimization function: Adam and SGD. SGD performed better on an extensive dataset due to its learning rate. Considering the size of this dataset is relatively small because we only need to detect parking signs instead of 20 or more different classes of objects, we use Adam.

4.2. Results

4.2.1. Dataset data

The training dataset that we used containing 4k parking sign data, of which 2k are parking sign images taken by human beings, and 2k of them are street view video frames from dash-cams. These videos cover four different areas: Tacoma, Boulder, Connecticut, and New York. We collected parking signs with different shapes (in Figure 8) and different types (in Figure 9). We also have variations of the same parking sign under complex road conditions, for example, light (in Figure 10), angle (in Figure 11), etc.

Figure 8: Parking Signs with Different Shape: circle, vertical, horizontal

Figure 9: Parking Signs with Different Types: parking signs for different group of vehicle type like loading, truck, taxicab, general payment, handicap

Figure 10: Parking Sign Under Different Light Condition

Figure 11: Parking Signs with Different Angle

Before training, we split our dataset two times. The first time we split the training stage and testing stage dataset at the rate of 9:1. Then for the training stage sub dataset, we did a second time split. The training stage subset was split into train and validation datasets at the rate of 9:1. Eventually, the dataset was split into Train, Validate, Test with sizes 3366, 400, and 425, respectively.

4.2.2. Weight Choose between yolov5s, yolov5m, yolov5l

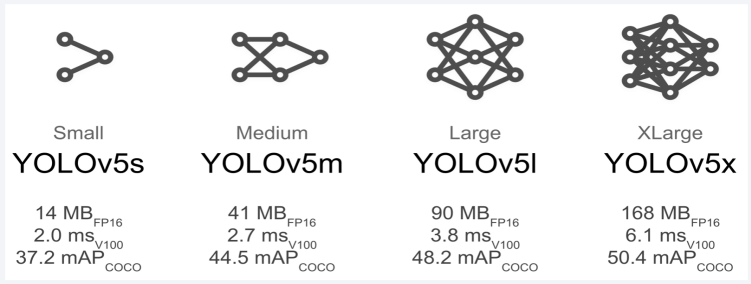

Figure 12: Yolo v5 Performance

Like most detection models, there is a tradeoff between speed and accuracy. A bigger size of the model has better accuracy but is relatively slow. Figure 12 shows how different sizes of Yolo v5 models perform. We only compare the first three models because Yolov5x has a size greater than 100 MB, which is too big to fit our purpose. In Table 1, we show how different sizes of Yolo v5 models perform on the parking sign dataset. According to Table 1, the accuracy AP@.5 of the three models is close. However, for the model size and speed,

Yolov5s has much better performance. Yolov5l has 6.5 times model size and 2.5 times detection time compared to the yolov5s. For the parking sign dataset, the minor accuracy increase is not worth trading so much detection time and model size. Therefore, we use yolov5s as the beginning weight to train the model in this thesis.

Table 1: Choose Model Size for Parking Sign Detection

Model	Pre-Trained	Dataset	Epoch	AP@.5	Model Size	Speed
Old	RetinaNet	V1	128	0.786	290MB	200ms/ 4.9FPS
New	yolov5s	V4	200	0.968	14.4MB	6.1ms/ 163FPS

4.3. Final model performance

Table 2: Test Result on V4 test Dataset

Model	Train on Dataset	Model Size	AP@.5[1]	Speed
yolov5s	V1[2]	14.4MB	0.831	5.4ms
yolov5m	V1	42.2MB	0.853	9.7ms
yolov5l	V1	93.7MB	0.858	13.7ms

After seven rounds of the training cycle, the self-trained model currently has good performance on the testing dataset. This thesis eventually trained a model with 14.4MB that can detect an image in 6.1ms with AP@.5 of 0.968. Compared to the RetinaNet[20], which trained on the old version (V1) dataset, the new model has a 23% increase on AP@.5 with only 5% of the old mode size. The old RetinaNet model did not reach the requirement of real-time, which is 30 frames per second. The old RetinaNet can only predict 5 frames per second, and the new Yolov5 model can predict 163 frames per second. The specific

[1] AP@.5: Mean Average Precision at IoU threshold 50%
[2] Dataset V1: 2112 Parking Sign Image collect and labeled by previous group member

performance data of the two models are shown in Table 2. We also include some exemplar detection results on complex parking sign images (i.e., Figure 13) and frames of a street view video (i.e., Figure 14), which further demonstrates our work in this thesis.

Figure 13: Complicated
Parking Sign Image Demo

Figure 14: Simple Video Frame Demo

5. Conclusion

In this thesis, we addressed a real-time parking sign detection problem to give real-time parking localization for further sign interpretation. Specifically, our model detects parking signs on the street using real-time video streaming of the street view as input, informing the parking laws for the present street. We accomplished two major objectives in this thesis: 1) creating a large parking sign dataset (over 4,000) that covers varieties of complicated street views; 2) using Yolov5 to train a real-time parking sign detection model with good performance and a modest model size on a customer parking sign dataset. Finally, we obtain a parking sign detection model with AP@.5 of 0.968 and a speed of 6.1ms per image. The model is only 14.4 MB, making it tiny enough to fit in mobile phones or self-driving automobiles.

5.1. Limitations and Future Directions

However, according to the detection result on our newest testing video, our current model still has some limitations. First, the model still has some false positives. For example, it has trouble distinguishing the parking sign and other road signs. Another weakness is that not all frames with a parking sign can be detected. According to the output video with the bounding box, which identifies the location of the parking sign object, the bounding box is flashing instead of stably appearing. This phenomenon shows that the model unsteadily predicts parking sign objects between frames. One potential solution is to use the nearby frame information.

Moreover, the model often fails to detect the object if another object shelters it. For example, if a parking sign is sheltered by a tree or even just sheltered by the tree's shadow, there will be a great chance that the parking sign will fail to be detected. However, if the previous frame and the later frame detect a parking sign with a similar size and location, it will be reasonable to predict that frame has a parking sign at the midpoint of these two locations. With this additional information from the adjacent frames, we can increase the object detection accuracy and may solve the bounding box flashing problem.

References

[1] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627–1645*, 2010.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

[3] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., "You Only Look Once: Unified, Real-Time Object Detection", *arXiv e-prints, 2015. arXiv:1506.02640*

[4] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M., "YOLOv4: Optimal Speed and Accuracy of Object Detection", *arXiv e-prints, 2020. arXiv:2004.10934*

[5] Girshick, R., "Fast R-CNN", *arXiv preprint*, Apr 2015. *arXiv:1504.08083*

[6] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. CSPNet: A new backbone that can enhance learning capability of cnn. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPR Workshop)*, 2020.

[7] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8759–8768, 2018.

[8] Joseph Redmonand, Ali Farhadi. YOLOv3: Anincremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[9] Nelson, Joseph, and Jacob Solawetz. "YOLOv5 Is Here: State-of-the-Art Object Detection at 140 FPS." *Roboflow*, Jun 10, 2020, blog.roboflow.com/yolov5-is-here/

[10] Rajput, M., "YOLO V5 Is Here! Custom Object Detection Tutorial with YOLO V5." Towards AI, 14 Jun 2020, pub.towardsai.net/yolo-v5-is-here-custom-object-detection-tutorial-with-yolo-v5-12666ee1774e

[11] Solawetz, J., "YOLOv5 New Version - Improvements And Evaluation" roboflow, 29 Jun, 2020 https://blog.roboflow.com/yolov5-improvements-and-evaluation/

[12] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao and Karol Zieba." End to End Learning for Self-Driving Cars", 2016; *arXiv:1604.07316.*

[13] Yolo v5: https://github.com/ultralytics/yolov5

[14] Mastromichalakis, S., "ALReLU: A different approach on Leaky ReLU activation function to improve Neural Networks Performance", *arXiv e-prints, 2020. arXiv:2012.07564*

[15] Li, J., "An Algorithm for Street Parking Sign Rule Generatio" *University of Washington Tacoma,* 2020

[16] Duran-Vega, M. A., Gonzalez-Mendoza, M., Chang-Fernandez, L., & Suarez-Ramirez, C. D. (2021). TYolov5: A Temporal Yolov5 Detector Based on Quasi-Recurrent Neural

Networks for Real-Time Handgun Detection in Video. *arXiv preprint arXiv:2111.08867.*

[17] Gu, Y., Wang, Q., & Qin, X. (2021). Real-time Streaming Perception System for Autonomous Driving. *arXiv preprint arXiv:2107.14388.*

[18] Qi, D., Tan, W., Yao, Q., & Liu, J. (2021). YOLO5Face: Why Reinventing a Face Detector. *arXiv preprint arXiv:2105.12931.*

[19] openvinotoolkit/cvat: https://github.com/openvinotoolkit/cvat

[20] Jian, Z., "Street Parking Sign Detection, Recognition and Trust System" *University of Washington Tacoma*, 2019

[21] Comaschi, F., Stuijk, S., Basten, T., & Corporaal, H. (2013, November). RASW: a run-time adaptive sliding window to improve viola-jones object detection. In 2013 *Seventh International Conference on Distributed Smart Cameras (ICDSC)* (pp. 1-6). IEEE.

[22] Li, J., Samrith, P., Guobadia, N., Hu, J., Chen, W. (June 2021) Automatic Street Parking Sign Reading. *Internet of Things, Ad Hoc and Sensor Networks Technical Committee Newsletter (IoT-AHSN TCN),* Vol. 1, No. 14. IEEE

[23] Humayun Irshad, Qazaleh Mirsharif, and Jennifer Prendki. Crowd sourcing based active learning approach for parking sign recognition. *arXiv preprint arXiv:1812.01081, 2018*.

[24] Faraji, P. H., Tohidypour, H. R., Wang, Y., Nasiopoulos, P., Ren, S., Rizvi, A., ... & Leung, V. C. (2021, September). Deep Learning based Street Parking Sign Detection and Classification for Smart Cities. *In Proceedings of the Conference on Information Technology for Social Good* (pp. 254-258).