Multiple Object Detection in Street Parking Signs for Smart Street Parking

Victor Chau (hqchau@uw.edu)
Supervised by Dr. Juhua Hu and Dr. Wei Cheng

***Abstract*—Automatic traffic infrastructure is essential for smart cities. Street parking sign detection and recognition are beneficial to autonomous driving and for current drivers, though it is a more challenging task than traffic sign detection due to its diversity and complexity. In this thesis, we introduce a deep learning based approach for street parking sign understanding through multiple object detection and explore multiple state-of-the-art models such as YOLOv5 and Swin Transformer for this task. Our experimentation showed that YOLOv5 is sufficiently fast and lightweight while achieving 98.3% mAP.**

I. Introduction

As technology advances at such an incredible rate, more and more metropolitan areas are transitioning to smart cities, and automatic traffic systems play a key role in those infrastructures. Automatic detection and recognition of street parking signs are essential not only for autonomous driving in the future but also for drivers in the present looking for parking in the middle of complex traffic systems on a daily basis. While street parking signs tend to convey similar messages, their styles are vastly different across cities and states in the United States. Furthermore, some signs can get complicated (see Fig. 1) through information stacking and distract even the most experienced drivers, potentially leading to dangerous situations or traffic congestion.

Traffic sign detection was an extensively researched topic in recent decades [2]. Though street parking sign detection and recognition is a similar task, it is a relatively unexplored problem. Street parking signs typically include multiple symbols with various texts as opposed to traffic signs which are designed to be fairly simple for quick user interpretation. And as mentioned, street parking signs are extremely diverse which makes its recognition task more challenging.

Recently, multiple developers proposed a method for street parking sign recognition [3]–[5] in two stages: sign detection then sign type classification. The authors used RetinaNet [6] to detect street parking signs from images taken by drivers. Then the cropped sign go through a series of SqueezeNet [7] classification models to determine if any No-Parking symbol or Arrows are presented on the sign. The main drawback of this framework is that its approach to sign symbol detection is rather inflexible. Expanding this method to detect one new symbol type would require extra effort into data collection and training for another SqueezeNet model, as well as putting extra computational load on the framework. In contrast, Faraji et al. [8] handled the street parking sign recognition task with just one single object detection model by assuming each sign

Fig. 1: A stack of parking signs in Los Angeles [1]

conveys one single parking rule of either No-Parking, Parking-Allowed, or No-Stopping. However, this assumption does not hold when a parking sign contains multiple symbols or divided into sections (see Fig. 1). Our approach to fully understand a parking sign is to extract as much information from it as possible with a scalable design.

In this thesis, we introduce a deep learning based method to simultaneously detect symbols and text located on street parking signs. This problem can be solved using state-of-the-art multiple object detection frameworks, such as YOLOv5 [9] and Swin Transformer [10]. A method compatible with smart systems must perform in real-time speed of at least 30 FPS and preferably lightweight. Our YOLOv5 model was able to run at 88.5 FPS while occupying less than 100 MB. Additionally, to address the lack of public data for street parking signs, we introduce our dataset of signs across multiple cities in the United States.

This thesis is structured as follows. Section II introduces a

brief development of object detection methods and examines some of the previous frameworks for street parking sign detection and understanding. Section III describes the detection pipeline that we worked on, a brief overview of the methods we chose, and the details of the street parking sign datasets. Section IV outlines the environment, the baseline model, and configurations we used to conduct experiments. We discuss the results in Section V with their implications and conclude the thesis in Section VI with discussions on the future directions.

II. Related Work

A. Object Detection

Object detection is an important problem of computer vision involving two tasks: (1) detect the presence and location of one or more objects in an image and (2) identify their type of object. In simpler terms, it answers the question "**what** objects are **where**?" Since the introduction of the first successful deep learning based architecture in 2014, object detection started to evolve rapidly. The field can be grouped into two genres: "one-stage detection" and "two-stage detection".

Two-stage object detection frameworks, as known as "coarse-to-fine" method, consist of two major phases: 1) Extract region of interests using proposal algorithms, and 2) Classify each proposed region by passing them through convolutional networks (CNNs) to extract their features. Leading of this architecture was Regions with CNN features (R-CNN) proposed by Girshick et al. [11]. The method used selective search [12] to propose regions of interest up to approximately 2,000 bounding boxes. Each is then resized to some fixed dimensions and fed into a CNN network to extract features. For region categorization, it used linear Support Vector Machine (SVM) classifiers. R-CNN achieved staggering performance at the time but suffered from extremely slow detection speed (14s per image with GPU). Nevertheless, R-CNN was a breakthrough in the object detection area in 2014 and remained as the heart of many two-stage detectors such as Fast R-CNN [13], Faster R-CNN [14], and Feature Pyramid Networks (FPN) [15].

One-stage detection, as the name suggested, completes the entire object detection process in one step by only passing the image through the predictive network once. The main method for this network is dividing the image into grids as anchors then simultaneously predicting bounding boxes and classification probabilities. The first method for this genre was the You Only Look Once (YOLO) [16], for which we will discuss more in detail in the System Overview section, followed by Single Shot multi-box Detector (SSD) [17] and RetinaNet [6]. In general, one-stage detectors are significantly faster (e.g. YOLO achieved 155 FPS) at the cost of accuracy.

Another architecture that is rising in popularity in the object detection sphere is Transformer [18], a type of deep neural network first introduced in 2017 based on the self-attention mechanism that was applied to the field of natural language processing (NLP). The self-attention mechanism learns the relationships between elements in a sequence, thus being able to generalize long-range relationships. This perk is valuable since most state-of-the-art computer vision techniques incorporate CNN which by design has localization inductive bias. However, the computational complexity for this mechanism is quadratic, and an image contains significantly more information than in a sentence. Thus, while having comparable performance to traditional CNN-based methods, Transformers struggle to reach real-time detection speed. This structure is under intensive research from the computer vision community, and in this thesis, we will examine one state-of-the-art Transformer model for our task as a baseline.

B. Parking Sign Detection and Understanding

To our best knowledge, not much work had done on the subject of object detection in street parking signs. The problem of traffic sign detection was the more popular interest. There were a few attempts to solve the street parking sign recognition and understanding.

In terms of parking sign recognition, [19] proposed a Linear SVM based framework to detect on-street parking signs. They made use of Google Street View API to collect street-level images in San Francisco for their training data. [20] adopted the same method to collect parking sign data but on a larger scale. They recognized the inefficiency of parsing Google Street View images for parking signs and developed an active learning method to solve this drawback. For parking sign detection, they evaluated SSD and YOLOv2 [21] which were the two popular deep learning based object detection frameworks at the time.

[3]–[5] presented a framework for on-street parking sign detection and understanding in the perspective of mobile application development, where users expect parking rules from snapshots of parking signs taken from their phones. For parking sign recognition, they suggested that RetinaNet would yield the best performance in terms of accuracy and speed for this task. To understand the context of each parking sign, text locations are recognized by a convolutional recurrent neural network (CRNN) before being fed into a parking rule generation algorithm. They treated the detection of important objects on a sign as a binary classification task and used SqueezeNet [7] as the model for no parking and arrow symbol detection. This means each new type of symbol would require collecting data, labeling, and training a new model - the process which demands a substantial amount of manual work. Moreover, increasing the number of models in the pipeline involves more storage and processing time. Since we know there are other types of symbols meaningful to parking rule generation other than No-Parking and Arrows, such as Tow-Away and Handicap, extending the object detection component of these frameworks proves to be a challenge.

Most recently, Faraji et al. [8] took a slightly different approach to street parking sign understanding. Instead of going through two phases of sign detection then interpretation, the authors combined them into one single step by assuming each sign conveys one parking rule and there are only three classes of signs: no-stopping, no-parking, and parking-allowed. They used YOLOv4 [22] as the model to detect signs in Vancouver,

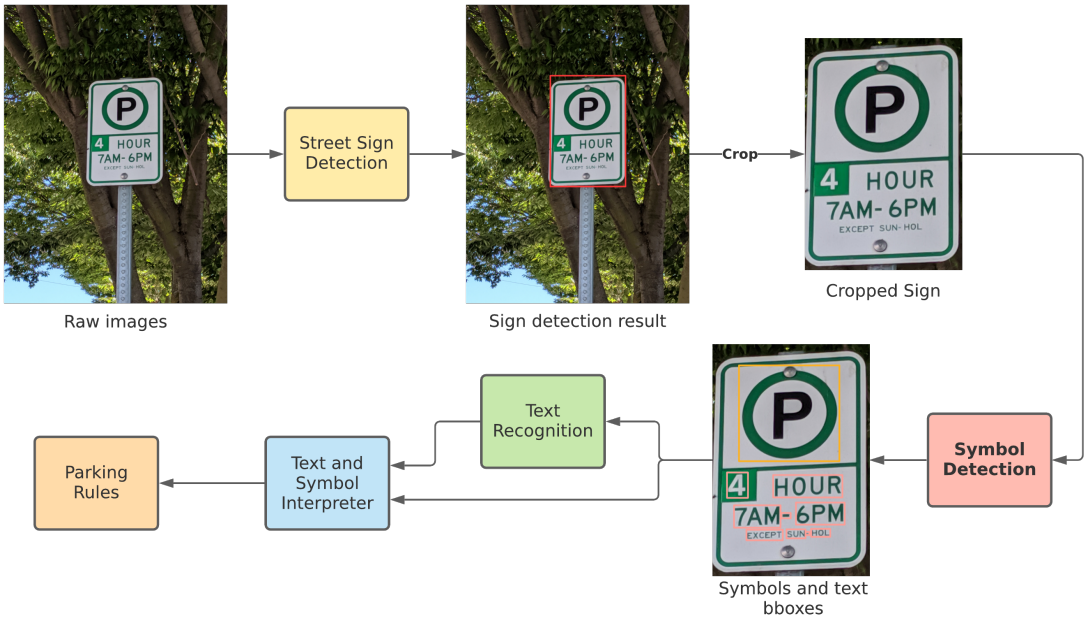

Fig. 2: A parking rule generation pipeline. This thesis focuses on the task of Symbol Detection.

Canada. Although the authors acknowledged the lack of text interpretation and mentioned that would be their next step for the project, their method would only perform well for simple signs. More often we encounter parking signs that contain more than one symbol or are divided into multiple sections, or both, which increase the complexity of the parking rule for one sign substantially.

In this thesis, given a parking sign, we aim to use one single model to handle significantly more types of symbols including text, without sacrificing storage and processing time.

III. Methodology

A. Detection Pipeline

We designed our parking sign symbol detection method with the intention for it to work within a parking generation pipeline similar to [3]–[5] which can be summarized in Fig. 2. Concretely, the input is a street image containing one or more parking signs taken by the user or extracted from a video feed. This image is then going through a street parking sign detection where the model will locate and crop out the sections that only contain a sign with minimal background. Each of the cropped signs is next fed to our symbol detection model to predict all relevant objects on the sign, and the results pass through the text and symbol interpreter where parking rules are finally generated.

From the pipeline, we see that the input for the symbol detection model is an image in the form of a three-dimensional vector representing $Width \times Height \times Color$. The output for each object detected includes the following information:

1) One integer signifies the object's class
2) One float for the prediction confidence score
3) Four floats for bounding box location of this object

The confidence score ranges from 0 to 1 and we use it to filter weak predictions which reduce the computational load for later stages in the pipeline. We discuss the details on how to calculate this value in the next section.

B. Architecture Overview: YOLOv5 [9]

In 2015, Redmon et al. [16] introduced YOLO, a one-stage detector that reframed the two-stage object detection into a single regression problem and end-to-end system from raw image pixels to bounding box and object class predictions.

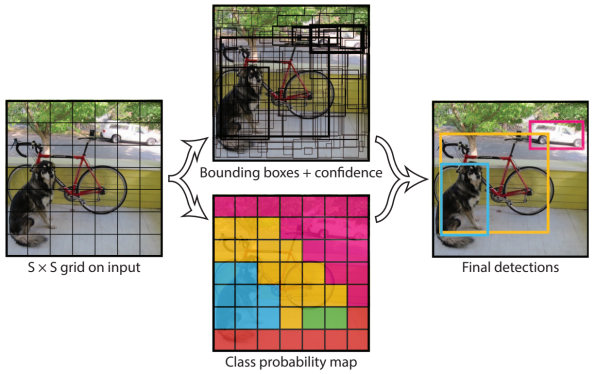

Fig. 3: Grid detection system of YOLO for $S = 7$ [16].

The object detection procedure of YOLO is called Unified Detection, where it overlays a grid of size $S \times S$ on top of the input image. Each cell in this grid is responsible for predicting two parameters: 1) B number of bounding boxes and confidence for these boxes and 2) the C class probabilities.

The confidence score reflects how confident the model thinks a cell is containing an object, regardless of its class. It is calculated by the product $P(\text{Object}) * IOU_{pred}^{truth}$ where $P(\text{Object})$ is either 0 or 1 indicating the object presence and

IOU_{pred}^{truth} is the intersection over union value between the predicted box and the ground truth.

Besides the confidence score, each bounding box prediction also consists of 4 other values: x, y, w, h. The coordinates (x, y) is the center of the box relative to the bounds of each grid cells, while the width-height pair (w, h) is relative to the whole image. An example to calculate these 4 prediction values is shown in Fig. 4.

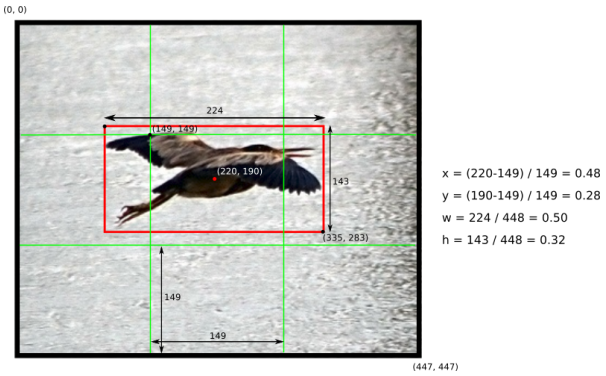

Fig. 4: Coordinates calculation example on a 3x3 grid [23].

Each class probability among C predictions, denoted $P(\text{Class}_i|\text{Object})$, is conditioned on the grid cell containing an object, meaning the value is valid if the grid contains an object. This probability is determined once for every grid cell, regardless of the number of B bounding boxes it is predicting, so all boxes in the same grid will share a common class.

Finally, YOLO uses Non-Maximum Suppression (NMS) to filter off bounding boxes that do not contain any objects and keep only the strongest predictions if two or more boxes overlap over a certain IOU threshold.

To arrive at the class confidence score for each bounding box, we multiply the class probability with the bounding box confidence score:

$$P(\text{Class}_i|\text{Object}) * P(\text{Object}) * IOU_{pred}^{truth}$$

which is one of the outputs we mentioned in the previous section.

The first version of YOLO was able to score sufficient accuracy while achieving more than real-time detection (155 FPS). Over the next five years, incremental improvements were made to this detection design and YOLOv5 is the latest adaptation.

The network architecture of YOLOv5 is shown in Fig. 5. YOLOv5's backbone to extract features incorporates cross-stage partial network (CSPNet) [25]. CSPNet solves the problem of duplicate gradient information, effectively reducing parameters of the model and computational bottleneck, resulting in smaller model size, faster training, and better inference speed. This is important for our purpose of embedding a real-time street parking recognition system into smart devices. Secondly, as part of YOLOv5's neck, path aggregation network (PANet) [26], which is an advanced version of FPN, was used to boost the propagation of low-level features, increasing the

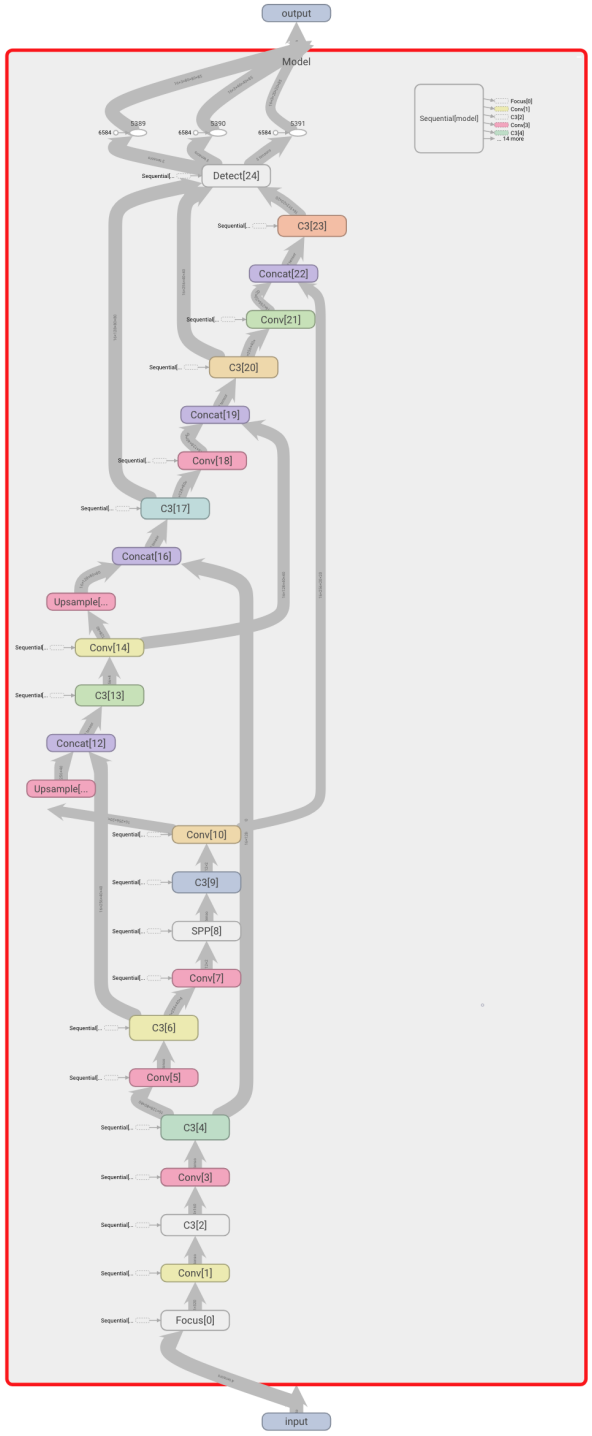

Fig. 5: Architecture of YOLOv5s (YOLOv5 small) visualized by TensorBoard [24].

localization accuracy of prediction bounding boxes. To further increase the prediction accuracy of the model as a whole, YOLOv5 also implements Mosaic data augmentation [22]. Finally, YOLOv5 utilizes the detection head mechanism of YOLOv3 [27] which predicts bounding boxes at 3 different scales of feature maps to achieve multi-scale prediction, al-

lowing the model to identify small, medium, or large objects. These perks give YOLOv5 the ability to accurately recognize small objects on a sign in cases of texts or low resolution images from video feeds.

C. Parking Sign Dataset

To the best of our knowledge, open-source data sets for street parking signs are extremely limited, thus, one contribution of this thesis is that we manually collected and annotated a data set for street parking signs covering different cities in US. The collected data consists of 3,369 images of street parking signs from large cities around the United States such as San Francisco, New York, Chicago, Seattle, etc. Images are either taken directly from mobile devices, on public internet archives or extracted from our dash-cam videos, so their qualities are varied. Some example parking signs can be observed in Fig. 6.

Fig. 6: Examples of collected parking signs.

From the collected data, we figure out 27 types of symbols that are useful for parking rule generation, most notably: text, no-parking, parking-allowed, handicap, tow-away, and arrows. Arrows are split into five classes, each indicating a different arrow direction: up, down, left, right, and bidirectional. Fig. 9 shows a comprehensive list with examples. There is a total of 28,142 labels (86.43% are texts) across all images in our collected data set. We divided the data with a 72:8:20 label ratio for train/validation/test. Labels are prioritized in the training set if there are not enough samples. Refer to Fig. 7 for the detailed class distribution.

We also form an extra test set from 821 images extracted from our dash-cam videos recorded in the city of Boulder Nevada, and the state of Connecticut. Both locations are not presented in the training data, and we will refer to this set as the "Boulder-CT" set. Similar to other samples obtained

class	train	val	test	total
no_parking	752	73	195	1020
text	17707	1790	4828	24325
handicap	50	5	16	71
parking	276	26	78	380
snow	6	1	1	8
no_stop	7	1	1	9
electric_vehicle	3	0	0	3
cleaning_vehicle	14	0	5	19
truck	8	0	3	11
tow_away	87	9	24	120
smart_phone	12	1	4	17
loading	12	1	4	17
fire_truck	3	0	0	3
pay_station	94	8	29	131
pay_with_hand	25	2	8	35
permit_star	5	1	1	7
permit_doc	1	0	0	1
bus	24	1	8	33
pay_by_phone	24	1	8	33
bicycle	9	0	4	13
scooter	9	0	4	13
arrow_right	492	48	130	670
arrow_left	517	53	141	711
arrow_bidir	289	33	82	404
arrow_up	31	2	12	45
arrow_down	26	2	11	39
taxi	4	0	0	4

Fig. 7: Distribution of classes.

Fig. 8: Examples in the Boulder-CT set. Their image resolutions are 168x189, 83x144, and 74x65, respectively. Second and third signs are resized to match the first one.

from dash-cam videos, their image resolutions are extremely low. The notable characteristic of the Boulder-CT set is that it contains multiple instances of the rounded sign with a down array (see Fig. 8 - left). It also includes extremely small handicap symbols (see Fig. 8 - right), though its type and placement on the sign are not unique.

IV. Experiments

A. Experiment Environment

We conducted our training and testing on a desktop PC with AMD Ryzen 9 5950X 16-core CPU, 32GB of RAM, and an RTX 3060 graphics card with 12 GB memory.

B. Baseline: Swin Transformer [10]

The previous implementation of Vision Transformer [28] while reaching top performance on computer vision tasks, still inherited the problem of quadratic computational complexity. Swin Transformer constructs a hierarchical representation of

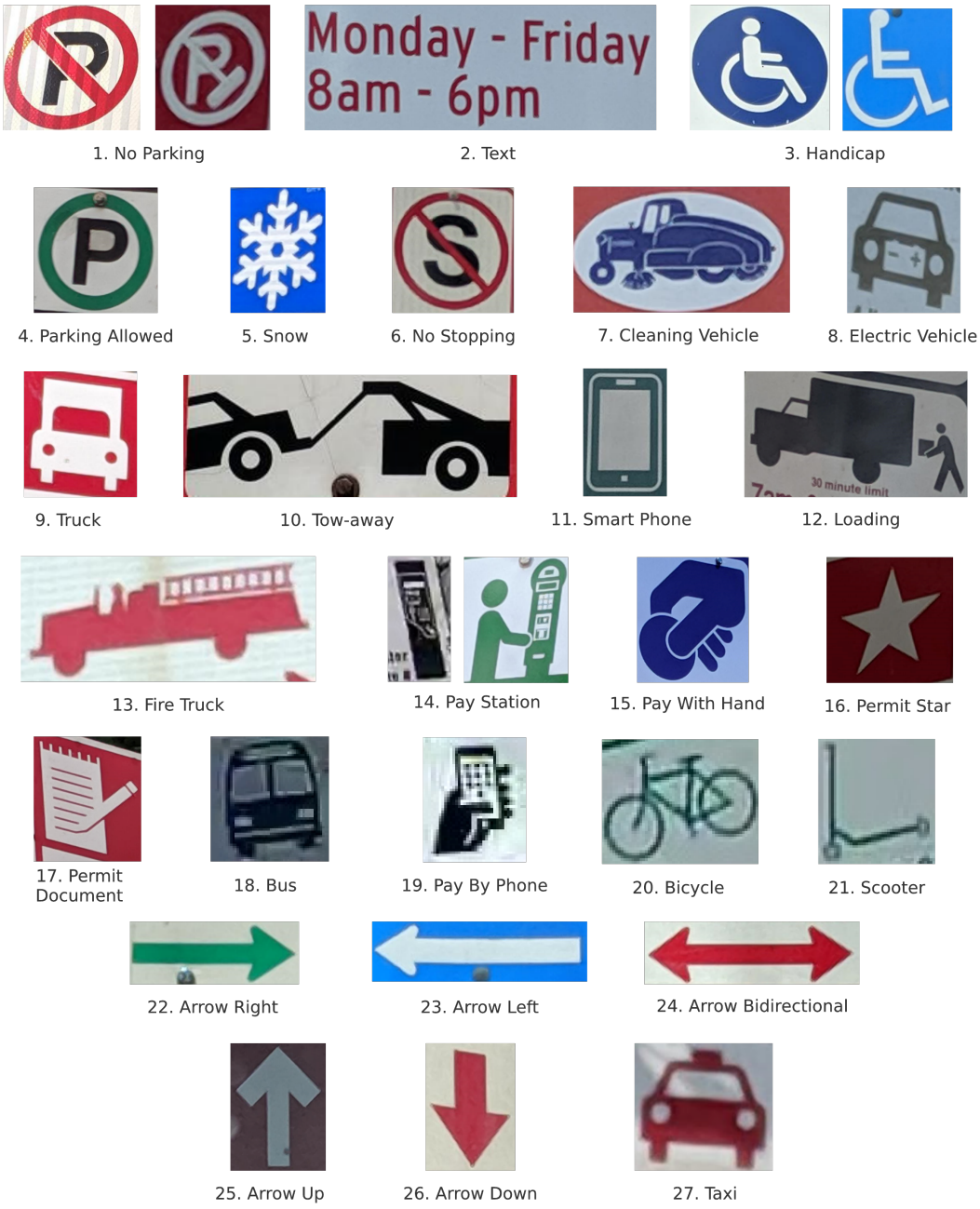

Fig. 9: Examples of all symbol classes in our parking sign recognition data.

the image by starting with small patches and gradually merging them in the process of feature extraction. To ensure global self-attention, it also implements a shifting window approach. Swin was able to bring the train and test complexity to almost linear while still producing strong detection results. We use Swin Transformer as the state-of-the-art baseline to measure the performance of our YOLOv5 models.

C. Training configurations

1) YOLOv5: We chose YOLOv5m6 and YOLOv6l6 (medium and large 6 version) [9] as the final models. The optimization algorithm adopts stochastic gradient descent (SGD). We set the network's initial learning rate to 0.01, the input of 640x640, momentum to 0.937, weight decay of 0.0005, and batch size of 16. The model was trained over 250 epochs.

2) Swin Transformer: We trained a Swin-T (tiny version) backbone and Cascade Mask R-CNN as the detection method, with mask heads removed since instance segmentation is not in the scope of our task. Weights are initialized with the pre-trained ImageNet-1K for the model provided by Swin's author [10]. We also adopted their training settings: multi-scale training of resizing the input of the shorter side between

480 and 800 while the longer side is at most 1333, and AdamW optimizer with weight decay of 0.05. Few parameter differences we implemented: initial learning rate is 0.00005, the batch size is 2, and 6x scheduler (72 epochs with the learning rate decayed at epoch 54 and 66 by a factor of 0.1).

We tried to use the largest batch size that our hardware allowed. Additionally, we augmented our data to avoid overfitting by adjusting brightness and saturation for both YOLOv5 and Swin. For YOLOv5, we also used perspective manipulation and random copy-paste. During our experiments, we observed that random horizontal flip augmentation worsened the performance of both models, specifically for only two classes arrow-left and arrow-right, possibly due to the fact that they are the mirror-image of each other.

V. Evaluation and Discussion

Considering the real-time application requirement, we evaluate these models with the following metrics: mean average precision (mAP) at the intersection over union (IoU) threshold of 0.5 and 0.5-0.95, recall, detection speed (in frame per second), and model size. The most important practical constraint is detection speed and we consider the ideal is at least 30 FPS.

We carried out two evaluation experiments. First, we used the test split of the training data to train the models and evaluate their performance on samples from the same pool of locations. Secondly, we use the Boulder-CT set to assess the trained models on signs in a brand new location.

TABLE I: Performance comparison of street parking symbol detection on the test split of the training data.

Model	Recall	$AP_{0.5}$	$AP_{.5:.95}$	Speed	Model Size
YOLOv5m6	0.980	0.983	0.859	88.5 FPS	68.4 MB
YOLOv5l6	0.976	0.992	0.892	56.8 FPS	147.0 MB
Swin-T	0.997	0.996	0.894	5.0 FPS	293.0 MB

We first compare the detection performance on the test split from the training data in Table I. We can observe that Swin Transformer returned better results than YOLOv5 in both precision and recall, but its detection speed is nowhere near the optimal requirement. The performance of the YOLOv5 medium is comparable to the large one while offering half the model size and an extra 30 FPS for detection speed, therefore, YOLOv5m6 is suitable to put in production of an application. Fig. 10 demonstrates some detection outputs from YOLOv5m6 model for various types of symbol. While Swin did not meet the practical requirement for real-time applications, its near-perfect recall is beneficial for future labeling of new data, which is a decently labor-intensive task.

TABLE II: Performance of street parking symbol detection on Boulder-CT dataset.

Model	Recall	$AP_{0.5}$	$AP_{0.5:0.95}$
YOLOv5m6	0.746	0.804	0.669
YOLOv5l6	0.747	0.805	0.616
Swin-T	0.784	0.775	0.580

Second, we aim to investigate the models' detection ability of new types or poor quality symbols which is common for signs extracted from video feeds, using the Boulder-CT test set, which is summarzied in Table II. It can be observed that when introduced to a new data containing signs from other cities, our models' performances drop substantially.

Fig. 10: Detection examples from YOLOv5m6.

TABLE III: Performance per class on Boulder-CT dataset.

Class	Labels	Recall		$AP_{0.5}$	
		YOLO	Swin	YOLO	Swin
no_parking	92	0.957	0.967	0.963	0.961
text	5303	0.880	0.904	0.923	0.853
handicap	**100**	**0.469**	**0.690**	**0.727**	**0.690**
tow_away	45	0.933	1.000	0.966	0.997
arrow_right	181	0.950	0.978	0.956	0.975
arrow_left	167	0.891	0.881	0.912	0.872
arrow_bidir	100	0.861	0.850	0.950	0.848
arrow_down	**117**	**0.034**	**0.000**	**0.041**	**0.000**

Finally, besides the overall performance, we also examine the detection performance for each different classes as shown in Table III. Note that we are using the medium model to represent YOLO and any class not included in the table did not present in the dataset. While the performance on most classes is acceptable for both models, they struggled to identify new variants in handicap and arrow_down classes as follows.

1) Handicap: The recall of Swin is 69%, about 20% higher than YOLO's, meaning that Swin can recognize more of the true labels in the dataset than YOLO. This behavior

is in line with Swin's tendency to return false positives. As a result, it was able to detect low-resolution symbols better than YOLOv5.

2) Arrow_down: Both models detected almost none of the new type of down arrow in the Boulder-CT dataset. While the image quality of the entire dataset is not high, some samples of this class are certainly visible with well-defined outlines, plus our models did not seem to have much problem with other symbols. This suggested that their ability to generalize symbols to sub-types from different cities leaves a lot to be desired.

VI. Conclusion

In this thesis, we proposed a method to extract meaningful symbols on street parking signs based on YOLOv5 multiple object detection framework and introduce a dataset for this task.

From our evaluations with Swin Transformer as the baseline, we found that YOLOv5 is able to produce a lightweight and fast model for mobile and real-time applications while still achieving 98% mAP, while Swin on the other hand is more beneficial for labeling tasks. Our second experiment suggested that both frameworks are not able to generalize parking symbols to account for different symbol variations but Swin is more suitable to detect extremely small objects. Though, our sample size for this test is still small and more experimentation is needed to confirm these hypotheses.

For future direction, our priority is to solve the imbalance of classes in our training dataset through data augmentation (such as designing a more purposeful copy-paste process to oversampling underrepresented classes) or other training techniques (e.g., transfer learning). We will also evaluate performance of text detection more extensively in future work since text is vital for generating parking rules. Finally, more data is needed to address the problem of ineffective symbol generalization from cities to cities across the United States.

References

[1] R/crappydesign - the parking signs in la are next level crap, 2019.

[2] Banhi Sanyal, Ramesh Kumar Mohapatra, and Ratnakar Dash. Traffic sign recognition: A survey. In *2020 International Conference on Artificial Intelligence and Signal Processing (AISP)*, pages 1–6. IEEE, 2020.

[3] Zhongyu Jiang. Street parking sign detection, recognition and trust system. 2019.

[4] Jiayu Li. An algorithm for street parking sign rule generation. 2020.

[5] Jiayu Li, Putthida Samrith, Nicole Guobadia, Juhua Hu, and Wei Cheng. Automatic street parking sign reading. *IEEE IOT-AHSN TC Newsletter*, 1(14):3–4, Jun 2021.

[6] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[7] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[8] Parnia Haji Faraji, Hamid Reza Tohidypour, Yixiao Wang, Panos Nasiopoulos, Simon Ren, Arash Rizvi, Cloris Feng, Mahsa T Pourazad, and Victor CM Leung. Deep learning based street parking sign detection and classification for smart cities. In *Proceedings of the Conference on Information Technology for Social Good*, pages 254–258, 2021.

[9] Glenn Jocher. Github - ultralytics/yolov5: Yolov5 in pytorch >onnx >coreml >tflite, 2021.

[10] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[12] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[13] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[14] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[15] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[19] Qazaleh Mirsharif, Théophile Dalens, Mehdi Sqalli, and Vahid Balali. Automated recognition and localization of parking signs using street-level imagery. In *Computing in Civil Engineering 2017*, pages 307–315. 2017.

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

[21] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[22] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[23] Mauricio Menegaz. Understanding yolo, Mar 2018.

[24] Glenn Jocher. Where is yolov5 paper? · issue 1333 · ultralytics/yolov5, 2021.

[25] 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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.

[26] Kaixin Wang, Jun Hao Liew, Yingtian Zou, Daquan Zhou, and Jiashi Feng. Panet: Few-shot image semantic segmentation with prototype alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9197–9206, 2019.

[27] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[28] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.