Challenge 1.1: Evaluating Perception Model in Autonomous Driving

24-784 Special Topics: Trustworthy AI Autonomy

Prof. D. Zhao

Due: Friday Mar 17th, 23:59:59 EST

- Your online version and its timestamp will be used for assessment.
- We will use Gradescope to grade. The link is on the panel of CANVAS. This homework is to be completed **individually**. Please abide by the ACADEMIC INTEGRITY POLICY on the syllabus.
- Submit **your_code.ipynb** for exercise 1 to Gradescope under **Challenge-1.1 Programming** and your solutions in **.pdf** format to **Challenge-1.1**. Insert the generated images and plots in the **.pdf**. We will randomly test your code and manually check all answers.
- Please familiarize yourself with Python programming, if you have never used it previously. We only accept Python for the coding of all homeworks. To that end, attending Python recitation bootcamp at the beginning of this course is highly recommended.
- We advise you to start with the assignment early. All the submissions are to be done before the respective deadlines. For information about available credits for late days, refer to the syllabus in CANVAS.
- The accompanying Google Colab Notebook is available at: https://colab.research.google.com/drive/1kIKhFnqaXE1jqYmgA_OGjIZ3kNmnbOIk?usp=share_link

1 Introduction

In this challenge assignment C1.1, you are required to form a team in 3 to tackle a real-world problem in autonomous driving.

Perception system is of crucial importance in autonomous driving, although the prosperity of deep learning has brought huge success in different tasks like image classification, semantic segmentation, and object detection, these deep neural networks are oftern vulnerable against adversarial attack. How to build a robust perception model under different conditions (e.g. different orientations, different illumination conditions, or even different attacks) is still an open questions in the frontier for autonomous driving researchers.

In P1, we've already seen an example where deep neural networks fail to classify the stop sign under PGD attack. In this challenge, we'll step further to see how these attack on the stop sign will function in the object detection task in the physical world.

The goals of this challenge are:

- Learn about the Carla simulator, understand the basic frameworks of SafeBench_v2.
- Implement the evaluation metrics for object detection.
- Evaluate the object detection model under the given textures:
  - A single-stage anchor-based object detectors (YOLOv5),
  - A two-stage object detector with a region proposal network and a recognition network (Faster RCNN).
- Evaluate the impact of the size and position of patch to the detection results.
- Design your customized patch that could successfully attack object detection model.

In the first phase of our challenge C1, you're required to **implement an important evaluation metric** of object detection algorithms.

2 Object Detection Evaluation

Object detection is the task of detecting instances of objects of a certain class within an image. Each detection produced by the model is a tuple with six items: $(x_1, y_1, x_2, y_2, conf, cls)$. The first four items are the vertices of the rectangle bounding box predicted by the model, $conf$ is the confidence of the prediction, and cls is the class of objects with the predicted bounding box. Notice that you may have **more than one detection in a single image**.

The state-of-the-art methods can be categorized into two main types: one-stage methods and two stage-methods. For the single-stage detector, we use You Only Look Once (YOLO) for the experiments. We're using YOLOv5 in our experiments. For the two-stage detector, we useFaster R-CNN for evaluation, you are welcome to check the papers and related blogs for more details in the reference section.

The evaluation metrics of Object Detection include Intersection-over-Union (IoU) and mean Average Precision (mAP). The computation of IoU metrics are demonstrated in Fig. 1.

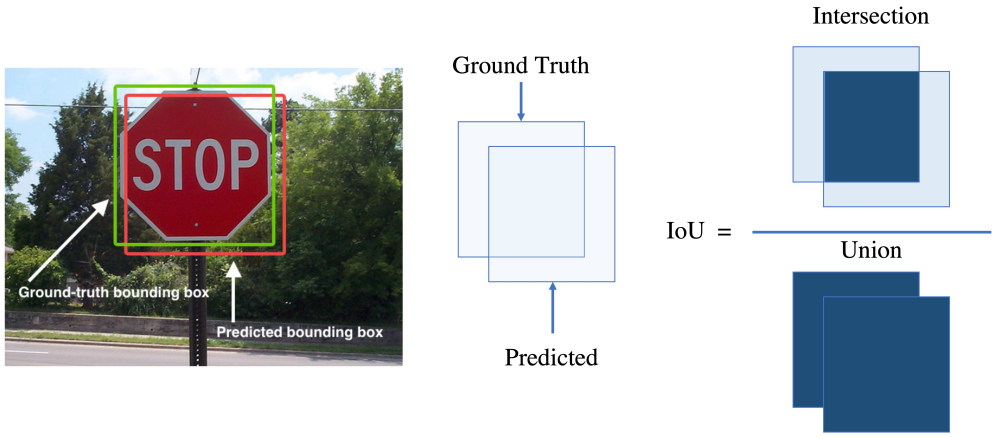

Figure 1: IoU metrics illustration, where the bounding box is given by four vertices (x_1, y_1, x_2, y_2) of a rectangle.

For each image, we'll have the ground truth labels as $(x_1^{gt}, y_1^{gt}, x_2^{gt}, y_2^{gt}, cls^{gt})$. After we get the IoU metrics for each detection result $(x_1, y_1, x_2, y_2, conf, cls)$ among all the images, we will get a set of predicted tuples $(IoU, conf, cls)$. We select a threshold of IoU, denoted as IoU_{thres}, then we can categorize all the prediction-label pairs into three categories:

- True Positive (TP): if the $cls = cls^{gt}$, and $IoU \geq IoU_{thres}$, then the prediction is considered as true positive.
- False Positive (FP): if the $cls = cls^{gt}$, but $IoU < IoU_{thres}$, then the prediction is considered as false positive.
- False Negative (FN): if there is cls^{gt}, but the model missed out at the current confidence level.

The true negative (TN) means the predictions that we managed to ignore do not correspond to any labeled object, which is not of our interests in the evaluation of object detection.

Then based on TP, FP, FN, we can define precision and recall as below:

$$\text{precision} = \frac{TP}{TP + FP} = \frac{TP}{\#(\text{predictions})} \tag{1}$$

$$\text{recall} = \frac{TP}{TP + FN} = \frac{TP}{\#(\text{ground truth})} \tag{2}$$

As the number of ground truth is fixed for a given dataset, the recall grows up monotonically increase as the confidence level goes down, while the precision may have some zig-zag as the confidence level goes down. We can then plot a precision-recall (PR) curve, denoted as $p(r)$, and compute the Average Precision by the areas under PR curve.

$$\text{AP} = \int_0^1 p(r)dr \approx \frac{1}{N}\sum_{i=0}^{N} p(\frac{i}{N}) \tag{3}$$

Since both precision and recall $\in [0, 1]$, AP $\in [0, 1]$. AP stands for the average precision at different recall(confidence) level. A higher AP means a higher precision of object detection model.

Next, let's see an example of how the Average Precision (AP) is derived. Assume we are having a table of prediction that is **sorted in descending order** w.r.t. the confidence level. Suppose we have 5 images, each images have a ground truth label as **Stop Sign**. In this case, we have #ground truth = 5. Then we'll have all the predictions in Table. 1. Then we can plot the PR curve by the precision-recall pairs in the last two columns in Fig.2.

Images	**Detection**	**Conf.**	**TP**	**FP**	**Acc TP**	**Acc FP**	**Precision**	**Recall**
Images 1	Stop Sign	0.95	1	0	1	0	1.00	0.20
Images 2	Car	0.90	0	1	1	1	0.50	0.20
Images 2	Person	0.88	0	1	1	2	0.33	0.20
Images 5	Stop Sign	0.80	1	0	2	2	0.50	0.40
Images 2	Stop Sign	0.75	1	0	3	2	0.60	0.60
Images 3	Stop Sign	0.70	1	0	4	2	0.67	0.80
Images 1	Car	0.50	0	1	4	3	0.57	0.80
Images 4	Stop Sign	0.60	1	0	5	3	0.63	1.00
Images 5	Car	0.45	0	1	5	4	0.56	1.00
Images 4	Person	0.30	0	1	5	5	0.50	1.00
Images 4	Tree	0.18	0	1	5	6	0.45	1.00

Table 1: An example of getting P-R Curve. The Acc TP and Acc FP are the accumulated true positive and accumulated false positive at the current confidence level.

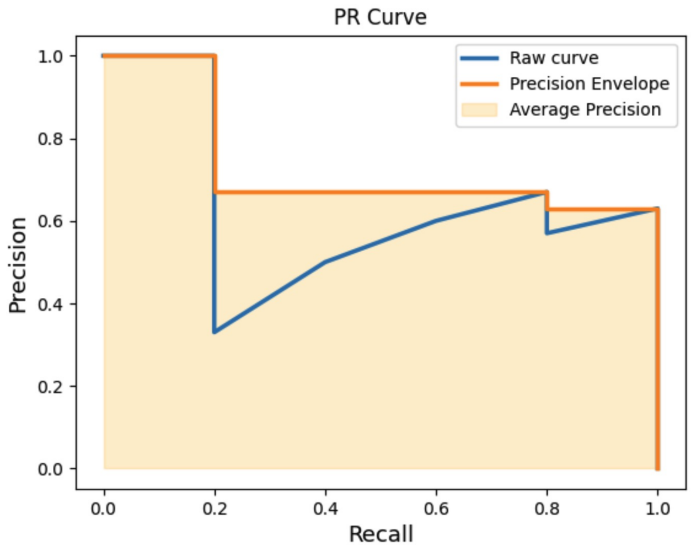

Figure 2: The raw PR Curve (directly connecting every point) is illustrated as blue line. In practice, we'll replace these zig-zag by replacing $p(r) = \max_{r' \geq r} p(r')$ in the orange curve. In this toy example, the Average Precision will be the areas under the P-R curve, which is colored as orange.

3 Problem Sets

3.1 Exercise 1: Implementation of AP Calculation

Suppose you are given a dictionary containing all the predictions from object detection model and their relationship with ground truth, similar to the example in Table. 1. There are a few important entries in the dictionary: (i) confidence scores; (ii) ground truth bounding box; (iii) predicted bounding box; (iv) predicted classes. You'll follow the introduction of Average Precision in Section 2.

Similar to the evaluation metrics in Common Objects in Context (COCO) dataset, please give the PR curve and corresponding AP at three different IoU threshold: 0.5:0.1:0.9, which means the threshold starts from 0.5, ends at 0.9 with a step size of 0.1. Plot all these five PR curves (**just the envelope curve instead of the raw curve**) into a single plot, and report all the five AP metrics for individual IoU threshold.

In your submission, please include the five P-R curve plots above. Also please give a brief introduction about how you compute the IoU loss. Then report your AP@IoU in a table, describe the trend you observe when the IoU threshold increase, and explain why.

4 Reference

1. AWS setup, https://www.youtube.com/watch?v=Yn-w3sfJil4.
2. SafeBench_v2: https://github.com/trust-ai/SafeBench_v2
3. Object Detection: https://www.jeremyjordan.me/object-detection-one-stage/
4. TorchVision models, https://pytorch.org/vision/0.8/models.html
5. YOLOv5: https://github.com/ultralytics/yolov5
6. Faster R-CNN: https://blog.paperspace.com/faster-r-cnn-explained-object-detection/