24784: Trustworthy AI Mar 15, 2024

Challenge 1

Name: Dalen Hsiao
Andrew ID: tungyuh
Collaborators: kaiyug

- **IoU Calculation**
  IoU is computed following these steps:

  1. The intersection area is computed by finding the overlapping area between the ground truth box and the predicted box, where we use the relatively small value of each entry of the input arrays to calculate the intersection area.
  2. The union area is computed by summing the area of the ground truth box and the predicted box while subtracting the union area.
  3. The IoU scores for each box are calculated by dividing the intersection area with the union area.

  The concept is illustrated as follow:

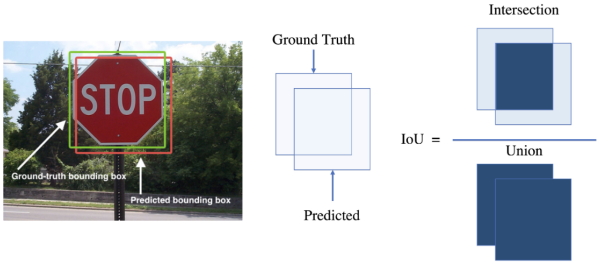


Figure 1: Calculation of IoU (source: Challenge 1 description)

- **Results Discussion**
  The calculations if precision and recall are as follow:

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

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

With varying IoU thresholds, we can expect that the higher the threshold, the likelihood of $FalsePositive$ instances appearing will decrease due to the increased confidence of the model. This occurs because the criterion for making positive predictions becomes stricter, essentially forcing the model to be more certain about all the positive predictions it makes. Therefore, at IoU thresholds greater than 0.7, we can start to observe a noticeable deviation between the Raw curves and the Orange curves, and the later is calculated by $p(r) = \max_{r' \leq r} p(r')$. As the

FP rate decreases, at the same level of recall rate, a lower precision could be expected and is understandable. From Table 1, we can see that the average precision (AP) trend reflects our initial assumption about the effect of the IoU threshold. Starting at an IoU threshold of 0.7, we observe a drop in AP, which continues as the threshold increases.

- P-R curves to each IoU

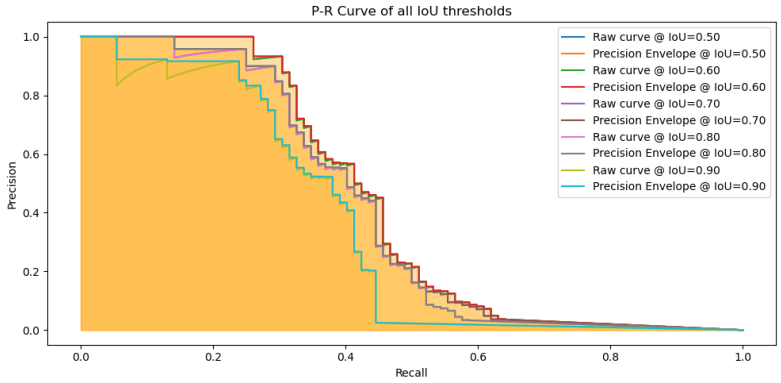


Figure 2: All P-R Curves

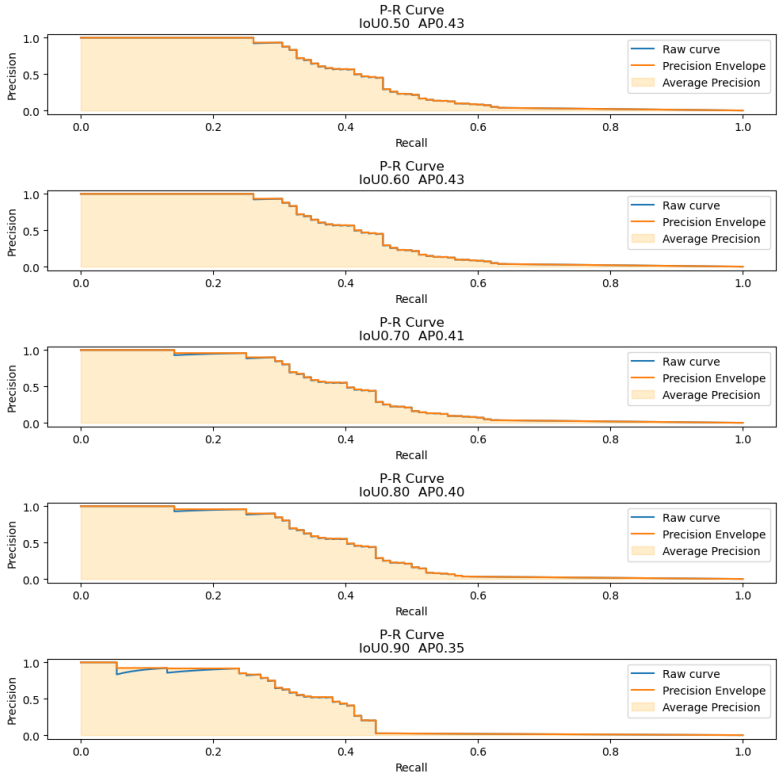


Figure 3: P-R Curves

- **IoU Threshold & Average Precision**

IoU Threshold	AP
0.50	0.4272559074361477
0.60	0.4272559074361477
0.70	0.40686292416368364
0.80	0.4018855902817313
0.90	0.3466670309329718

Table 1: IoU Threshold vs AP

C1.1 Evaluation on Object Detection

Introduction

- Task: Evaluate the average precision of the data collected from SafeBench.
- Submission: the plotted P-R curve, as well as the average precision at different IoU threshold level.

In [ ]:
```
import joblib
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
```

In [ ]:
```
# !git clone https://github.com/HenryLHH/24784-c1-data.git
# !cp /content/24784-c1-data/results.pkl ./
# !rm -r /content/24784-c1-data/
```

Load the Detection Results from SafeBench_v2

In [ ]:
```
inputs = joblib.load('results.pkl')  # move the files to your current folder
inputs.keys() # dict_keys(['image_id', 'predicted_class', 'ground_truth_bbox',
```

Out[ ]:
```
dict_keys(['image_id', 'predicted_class', 'ground_truth_bbox', 'predicted_bbo
x', 'conf_scores', 'num_labels'])
```

Compute the IoU

In [ ]:
```
def box_area(box):
    """ Compute the box area, given all the vertices
    # Arguments
        box:  (N,4) ndarray
    # Returns
        areas: (N, ) ndarray
    """
    (x1, y1, x2, y2) = box.T
    areas =  np.abs((x1-x2) * (y1-y2)) # TODO, compute the rectangle areas base
    return areas # (N, ) ndarray

def box_iou(box1, box2):
    '''
      Compute the iou between box1 and box2
      ONLY consider the SINGLE ground truth, which is a simplified case
      box1: (N, 4) ndarray
      box2: (N, 4) ndarray
      return: (N, ) iou scores
    '''
    eps = 1e-7

    a1, a2 = np.split(box1, 2, axis=1) # ground truth boxes
    b1, b2 = np.split(box2, 2, axis=1) # predicted boxes
```

```
    # intersections' boundaries
    x1 = np.maximum(a1, b1)
    x2 = np.minimum(a2, b2)

    inter =  np.abs((x1[:,0] - x2[:,0])*(x1[:,1] - x2[:,1]))
    union =  box_area(box1) + box_area(box2) - inter
    return inter / (union + eps)
```

In [ ]:
```
# TODO: get the box iou scores from the function you implemented above
iou_scores = box_iou(inputs['ground_truth_bbox'], inputs['predicted_bbox'])

# Build your dataframe from the dictionary
input_dict = {
        'image_id': inputs['image_id'],
        'predicted_class': inputs['predicted_class'],
        'conf_scores': inputs['conf_scores'],
        'iou_scores': iou_scores,
      }
df = pd.DataFrame.from_dict(input_dict)
df
```

Out[ ]:

	image_id	predicted_class	conf_scores	iou_scores
0	0	None	-1.000000	-1.001914
1	1	car	0.671808	-1.223964
2	1	car	0.563249	-1.642660
3	1	car	0.452749	0.800520
4	1	car	0.428004	0.229486
...	...	...	...	...
1631	91	train	0.219339	0.680153
1632	91	bus	0.173378	0.722757
1633	91	person	0.101061	0.203605
1634	91	sink	0.088729	0.044596
1635	91	dining table	0.073306	0.030152

1636 rows × 4 columns

In [ ]:
```
df = df[df.conf_scores >= 0] # drop all the frames without detection (conf sco

# get all the detection results of stop signs
df_stopsign = df.loc[(df.predicted_class=='stopsign')]
df_stopsign
```

Out[ ]:

	image_id	predicted_class	conf_scores	iou_scores
160	7	stopsign	0.071092	0.812147
345	15	stopsign	0.052572	0.823549
712	33	stopsign	0.110136	0.814429
731	34	stopsign	0.159796	0.844884
772	36	stopsign	0.128940	0.850620
...	...	...	...	...
1606	88	stopsign	0.119673	0.025573
1610	89	stopsign	0.997623	0.932374
1622	90	stopsign	0.995437	0.923847
1628	90	stopsign	0.052746	0.032057
1629	91	stopsign	0.995099	0.614191

66 rows × 4 columns

Compute the Avereage Precision

In [ ]:
```
def interp_ap(recall, precision, method = 'interp'):
    """ Compute the average precision, given the recall and precision curves
    # Arguments
        recall:    The recall curve (list)
        precision: The precision curve (list),
        methods: 'continuous', 'interp'
    # Returns
        Average precision, precision curve, recall curve
    """

    # TODO: Append sentinel values to beginning and end
    # Recall should start with 0.0 and end with 1.0
    # Precision should start with 1.0 and end with 0.0

    appended_recall = np.concatenate(([0.0], recall, [1.0]))
    appended_prec_input = np.concatenate(([1.0], precision, [0.0]))

    # Compute the precision envelope
    appended_prec = np.flip(np.maximum.accumulate(np.flip(appended_prec_input)

    # Integrate area under curve
    if method == 'interp':
        x = np.linspace(0, 1, 101)  # 101-point interp (COCO)
        ap = np.trapz(np.interp(x, appended_recall, appended_prec), x)      # g
    else:  # 'continuous'
        i = np.where(appended_recall[1:] != appended_recall[:-1])[0]  # points
        ap = np.sum((appended_recall[i + 1] - appended_recall[i]) * appended_p

    return ap, appended_prec_input, appended_prec, appended_recall
```

In [ ]:
```
def compute_ap(df, num_gt, iou_thres):
    """ Compute the average precision, given the recall and precision curves
```

```
    # Arguments
        df:         DataFrame inputs containing predicted classes, iou_scores,
        num_gt:     The precision curve (list),
        iou_thres:  IoU threshold of True Positive (TP) detection
    # Returns
        Average precision, precision curve, recall curve
    """
    df = df.sort_values(by='conf_scores', ascending=False)
    tp_fp = np.arange(1, len(df)+1, 1)
    tp = ((df['iou_scores'] >= iou_thres) & (df['predicted_class'] == 'stopsig
    precision = tp / tp_fp      # array (N, )
    recall = tp / num_gt        # array (N, )

    ap, prec_raw, prec, recall = interp_ap(recall, precision)           # get

    return ap, prec_raw, prec, recall
```

Visualization

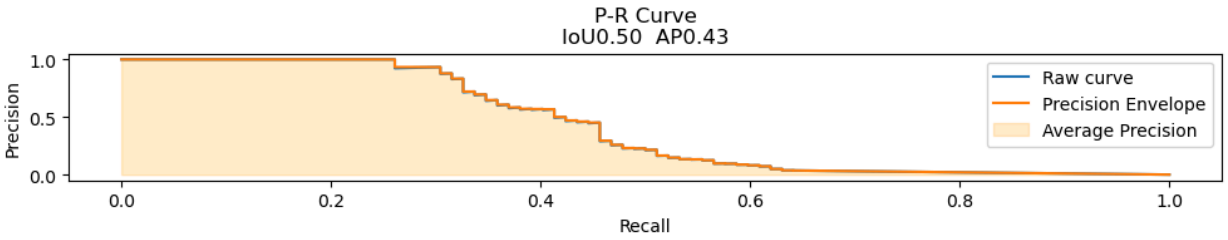
In [ ]:
```
def plot_pr_curves(recall, prec_raw, prec, title = None):
    """ Visualize the P-R curves
        Visualize the areas under P-R curves
    """
    # plt.figure()
    plt.plot(recall, prec_raw)
    plt.plot(recall, prec)
    plt.fill_between(recall, np.zeros_like(recall), prec, color='orange', alpha
    plt.legend(['Raw curve', 'Precision Envelope', 'Average Precision'])
    plt.title('P-R Curve' + title)
    plt.xlabel('Recall')
    plt.ylabel('Precision')
def plot_all_pr_curves(recall, prec_raw, prec, threshold):
    """ Visualize the P-R curves
        Visualize the areas under P-R curves all in one plot
    """
    # plt.figure()
    plt.plot(recall, prec_raw, label='Raw curve @ IoU={:.2f}'.format(threshold
    plt.plot(recall, prec, label = 'Precision Envelope @ IoU={:.2f}'.format(th
    plt.fill_between(recall, np.zeros_like(recall), prec, color='orange', alpha
    plt.legend(['Raw curve', 'Precision Envelope', 'Average Precision'])
    # plt.title('P-R Curve')
    plt.xlabel('Recall')
    plt.ylabel('Precision')
```

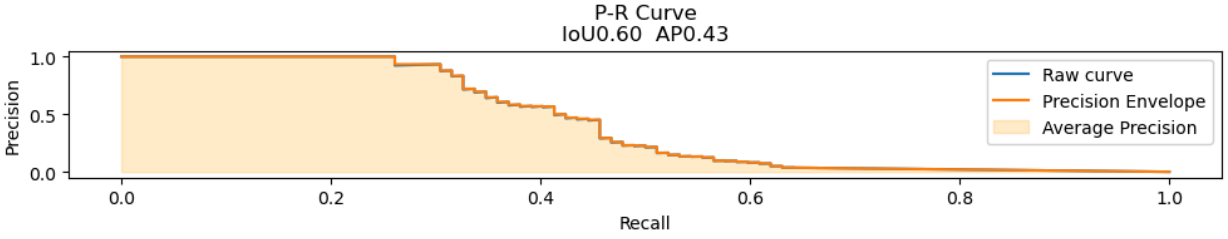
Execute the AP calculation

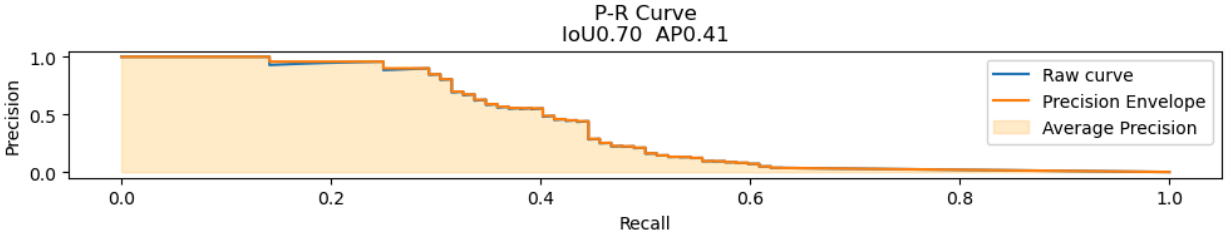
In [ ]:
```
# Traverse over IoU threshold, get AP@[0.5:0.1:0.9]
plt.figure(figsize=(10, 10))
logs = {}
for i, iou_thres in enumerate(np.arange(0.5, 1.0, 0.1)):
    ap, prec_raw, prec, recall = compute_ap(df, inputs['num_labels'], iou_thres
    print('IoU Threshold: {:.2f}'.format(iou_thres), '  |    AP@{:.2f}'.format
    logs['IoU Threshold: {:.2f}'.format(iou_thres)] = ap
    plt.subplot(len(np.arange(0.5, 1.0, 0.1)), 1, i+1)
    plot_pr_curves(recall, prec_raw, prec, title="\n" + f"IoU{iou_thres:.2f}  A
plt.tight_layout()
```

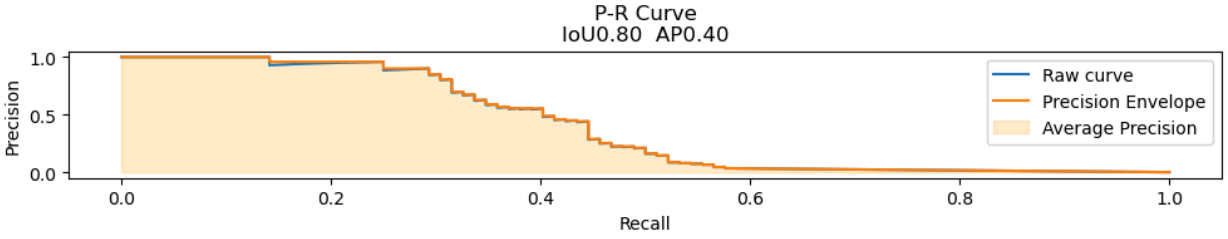
```
plt.show()

# plotting all the results in the same plot
plt.figure(figsize=(10, 5))
for i, iou_thres in enumerate(np.arange(0.5, 1.0, 0.1)):
    ap, prec_raw, prec, recall = compute_ap(df, inputs['num_labels'], iou_thres
    print('IoU Threshold: {:.2f}'.format(iou_thres), '  |    AP@{:.2f}'.format
    logs['IoU Threshold: {:.2f}'.format(iou_thres)] = ap
    plot_all_pr_curves(recall, prec_raw, prec, threshold=iou_thres)
plt.legend()
plt.title('P-R Curve of all IoU thresholds')
plt.tight_layout()
plt.show()
```

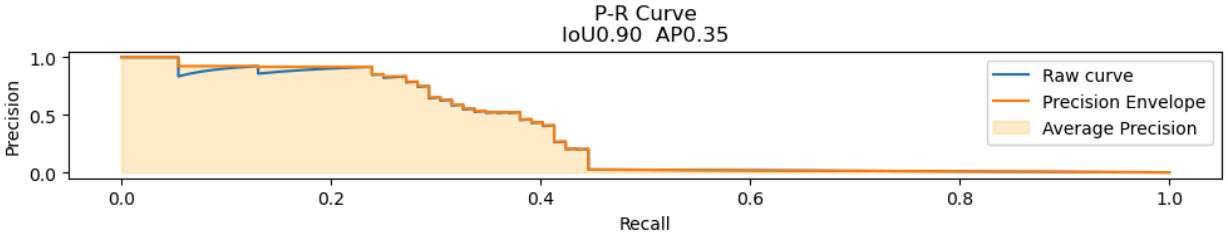
```
IoU Threshold: 0.50   |    AP@0.50 0.4272559074361477
IoU Threshold: 0.60   |    AP@0.60 0.4272559074361477
IoU Threshold: 0.70   |    AP@0.70 0.40686292416368364
IoU Threshold: 0.80   |    AP@0.80 0.4018855902817313
IoU Threshold: 0.90   |    AP@0.90 0.3466670309329718
```



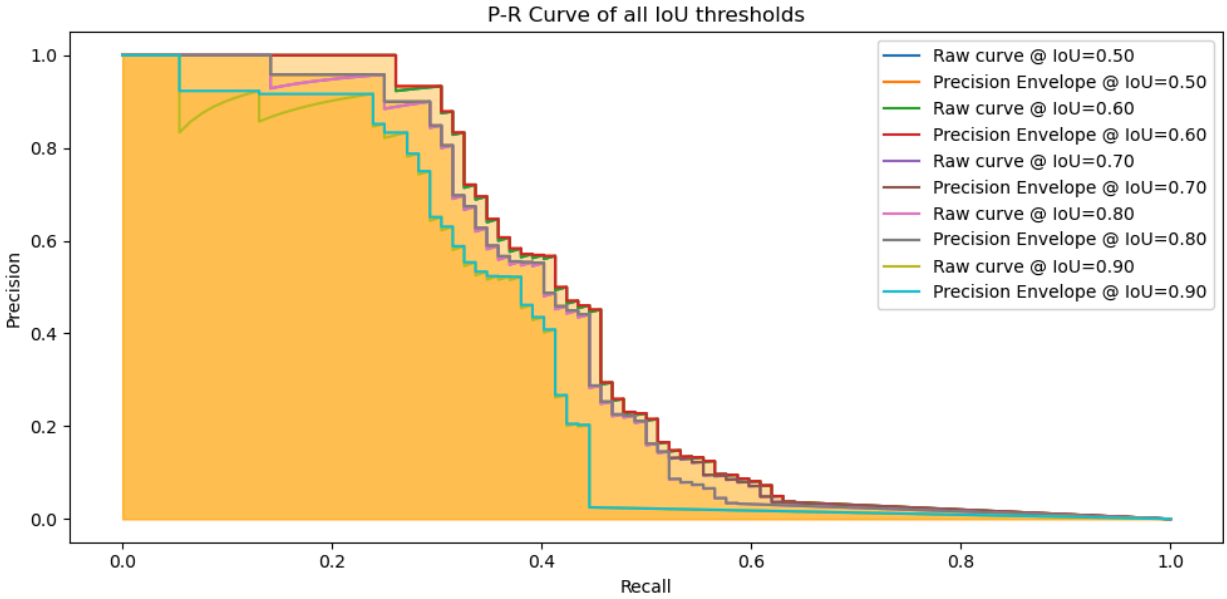








```
IoU Threshold: 0.50   |    AP@0.50 0.4272559074361477
IoU Threshold: 0.60   |    AP@0.60 0.4272559074361477
IoU Threshold: 0.70   |    AP@0.70 0.40686292416368364
IoU Threshold: 0.80   |    AP@0.80 0.4018855902817313
IoU Threshold: 0.90   |    AP@0.90 0.3466670309329718
```



In [ ]:
```
import json
with open('logs.json', 'w') as f:
    json.dump(logs, f)
```