abkditpyi

March 30, 2024

1 Challenge 2: Evaluating Perception Model in SafeBench

1.1 Team: *Paradox*

1.2 Teammates: 'kjamunap' , 'sriramna' , 'leonli'

1.3 Click here! for GoogleDrive link for the videos and images obtained

1.4 Exercise 1: Different Object Detection Models

1.4.1 Object Detection Models:

1. YOLOv5
2. Faster R-CNN

1.4.2 Textures:

- stopsign.jpg
- stopsign 1.jpg
- stopsign 2.jpg

Table that gives the AP@[0.5:0.05:0.95] results (with mean AP and IoU on 4 different scenarios) for 2 object detection models (YOLOv5 and Faster R-CNN) under 3 textures (stopsign.jpg, stopsign_1.jpg, stopsign_2.jpg)

Model	Texture	mean average precision (mAP)	Mean IoU
YOLOv5	stopsign.jpg	0.4927, 0.6929, 0.7879, 0.7961	0.7501, 0.8299, 0.8771, 0.8785
Faster R-CNN	stopsign.jpg	0.2126, 0.4034, 0.5526, 0.5282	0.3667, 0.4291, 0.5998, 0.5827
YOLOv5	stopsign 1.jpg	0.3687, 0.5249, 0.7265, 0.6877	0.5595, 0.5956, 0.8472, 0.8234

Model	Texture	mean average precision (mAP)	Mean IoU
Faster R-CNN	stopsign 1.jpg	0.1802, 0.3577, 0.3842, 0.3648	0.3331, 0.3976, 0.4296, 0.4241
YOLOv5	stopsign 2.jpg	0.3757, 0.5479, 0.7276, 0.6592	0.5437, 0.6398, 0.8453, 0.7702
Faster R-CNN	stopsign 2.jpg	0.1873, 0.3432, 0.5541, 0.4045	0.3483, 0.4053, 0.6826, 0.4992

[86]:
```
yolo0 = [0.7501, 0.8299, 0.8271, 0.8085]   # YOLOv5, stopsign.jpg
rcnn0=[0.3667, 0.4291, 0.5998, 0.5827]  # Faster R-CNN, stopsign.jpg
yolo1=[ 0.5595, 0.5956, 0.8472, 0.8234]   # YOLOv5, stopsign 1.jpg
rcnn1= [0.3331, 0.3976, 0.4296, 0.4241]   # Faster R-CNN, stopsign 1.jpg
yolo2=[0.5437, 0.6398, 0.8453, 0.7702]   # YOLOv5, stopsign 2.jpg
rcnn2=[0.3483, 0.4053, 0.6826, 0.4992]   # Faster R-CNN, stopsign 2.jpg

# Calculate the average of the mean values
average_yolo0 = sum(yolo0) / len(yolo0)
average_yolo1 = sum(yolo1) / len(yolo1)
average_yolo2 = sum(yolo2) / len(yolo2)
average_rcnn0 = sum(rcnn0) / len(rcnn0)
average_rcnn1 = sum(rcnn1) / len(rcnn1)
average_rcnn2 = sum(rcnn2) / len(rcnn2)

# Print the result
print(f"Average of mean values yolo_0:, {average_yolo0:.2f}")
print(f"Average of mean values yolo_1:, {average_yolo1:.2f}")
print(f"Average of mean values yolo_2:, {average_yolo2:.2f}")
print(f"Average of mean values rcnn_0:, {average_rcnn0:.2f}")
print(f"Average of mean values rcnn_1:, {average_rcnn1:.2f}")
print(f"Average of mean values rcnn_2:, {average_rcnn2:.2f}")
```

```
Average of mean values yolo_0:, 0.80
Average of mean values yolo_1:, 0.71
Average of mean values yolo_2:, 0.70
Average of mean values rcnn_0:, 0.49
Average of mean values rcnn_1:, 0.40
Average of mean values rcnn_2:, 0.48
```

[87]:
```
import numpy as np
```

```python
# Calculate the standard deviation for each set of values
std_yolo0 = np.std(yolo0)
std_rcnn0 = np.std(rcnn0)
std_yolo1 = np.std(yolo1)
std_rcnn1 = np.std(rcnn1)
std_yolo2 = np.std(yolo2)
std_rcnn2 = np.std(rcnn2)

# Print the results
print(f"Standard deviation of YOLOv5, stopsign.jpg: {std_yolo0:.4f}")
print(f"Standard deviation of Faster R-CNN, stopsign.jpg: {std_rcnn0:.4f}")
print(f"Standard deviation of YOLOv5, stopsign 1.jpg: {std_yolo1:.4f}")
print(f"Standard deviation of Faster R-CNN, stopsign 1.jpg: {std_rcnn1:.4f}")
print(f"Standard deviation of YOLOv5, stopsign 2.jpg: {std_yolo2:.4f}")
print(f"Standard deviation of Faster R-CNN, stopsign 2.jpg: {std_rcnn2:.4f}")
```

```
Standard deviation of YOLOv5, stopsign.jpg: 0.0321
Standard deviation of Faster R-CNN, stopsign.jpg: 0.0993
Standard deviation of YOLOv5, stopsign 1.jpg: 0.1298
Standard deviation of Faster R-CNN, stopsign 1.jpg: 0.0383
Standard deviation of YOLOv5, stopsign 2.jpg: 0.1163
Standard deviation of Faster R-CNN, stopsign 2.jpg: 0.1268
```

1.5 Mean and Standard Deviation for YOLOv5

1. **Std. Dev.**

Texture	YOLOv5 Std. Dev.	Faster R-CNN Std. Dev.
stopsign.jpg	0.0522	0.0993
stopsign 1.jpg	0.1298	0.0383
stopsign 2.jpg	0.1163	0.1268

2. **MEAN**

Texture	YOLOv5 Average Mean	Faster R-CNN Average Mean
stopsign.jpg	0.80	0.49
stopsign 1.jpg	0.71	0.40
stopsign 2.jpg	0.70	0.48

From this comparison, it's seen that YOLOv5 generally outperforms Faster R-CNN in terms of average mean precision across the different textures.

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

1.6 Exercise 2: Customized Patch Attack

We have used adversarial attack patches of texture: typewriter of size 198x197 and texture: pager mobile of size 200x200.Then placing the customized texture patch strategically within the stop sign area of a 512x512 input texture image. The center of the stop sign is typically around (310, 310) pixels in a 512x512 image. After designing the customized texture patch, evaluate its effectiveness by testing it with both YOLO-v5 and Faster R-CNN object detection models. Use drive-by testing to simulate real-world detection scenarios and record the detection results.

1.6.1 Results obtained:

Model	Texture	mean average precision (mAP)	Mean IoU
YOLOv5	typewriter	0.3842, 0.6332, 0.7452, 0.6778	0.5686, 0.7779, 0.8561, 0.8092
Faster R-CNN	typewriter	0.1908, 0.3597, 0.4978, 0.4092	0.3420, 0.4072, 0.5535, 0.4654
YOLOv5	pager mobile	0.4714, 0.6776, 0.7738, 0.7271	0.7046, 0.7977, 0.8724, 0.8486
Faster R-CNN	pager mobile	0.1577, 0.3207, 0.4082, 0.3566	0.3432, 0.4085, 0.5732, 0.4753

1.6.2 Videos is linked in google drive above

- Open Exercise_2 folder and you will find stopsign image for texture typewriter and pager-mobile
- Video will named as yolo_for_* mp4 (* :texture)

[88]:
```
yolo_type = [0.3842, 0.6332, 0.7452, 0.6778]   # YOLOv5, texture:typewriter.jpg
rcnn_type=[0.1908, 0.3597, 0.4978, 0.4092]  # Faster R-CNN, texture:typewriter.
 ↪jpg
yolo_mob=[ 0.4714, 0.6776, 0.7738, 0.7271]   # YOLOv5, texture:pager mobile.jpg
rcnn_mob= [0.1577, 0.3207, 0.4082, 0.3566]   # Faster R-CNN, texture:pager␣
 ↪mobile.jpg

# Calculate the average of the mean values
average_yolo_type = sum(yolo_type) / len(yolo_type)
average_rcnn_type = sum(rcnn_type) / len(rcnn_type)
average_yolo_mob = sum(yolo_mob) / len(yolo_mob)
```

```
average_rcnn_mob = sum(rcnn_mob) / len(rcnn_mob)

# Print the result
print(f"Average of mean values YOLOv5, texture:typewriter :, {average_yolo_type:
  ↪.2f}")
print(f"Average of mean values Faster R-CNN, texture:typewriter :,␣
  ↪{average_rcnn_type:.2f}")
print(f"Average of mean values YOLOv5, texture:pager mobile :,␣
  ↪{average_yolo_mob:.2f}")
print(f"Average of mean values Faster R-CNN, texture:pager mobile :,␣
  ↪{average_rcnn_mob:.2f}")
```

```
Average of mean values YOLOv5, texture:typewriter :, 0.61
Average of mean values Faster R-CNN, texture:typewriter :, 0.36
Average of mean values YOLOv5, texture:pager mobile :, 0.66
Average of mean values Faster R-CNN, texture:pager mobile :, 0.31
```

[89]:
```
std_yolo_type = np.std(yolo_type)
std_rcnn_type = np.std(rcnn_type)
std_yolo_mob = np.std(yolo_mob)
std_rcnn_mob = np.std(rcnn_mob)
# Print the results
print(f"Standard deviation of YOLOv5, texture:typewriter: {std_yolo_type:.4f}")
print(f"Standard deviation of Faster R-CNN, texture:typewriter: {std_rcnn_type:.
  ↪4f}")
print(f"Standard deviation of YOLOv5, texture:pager mobile: {std_yolo_mob:.4f}")
print(f"Standard deviation of Faster R-CNN, texture:pager mobile: {std_rcnn_mob:
  ↪.4f}")
```

```
Standard deviation of YOLOv5, texture:typewriter: 0.1364
Standard deviation of Faster R-CNN, texture:typewriter: 0.1118
Standard deviation of YOLOv5, texture:pager mobile: 0.1154
Standard deviation of Faster R-CNN, texture:pager mobile: 0.0937
```

1.7 Mean and Standard Deviation for YOLOv5 and Faster R-CNN

1. **Std. Dev.**

Texture	YOLOv5 Std. Dev.	Faster R-CNN Std. Dev.
Typewriter	0.1364	0.1118
pager mobile	0.1154	0.0937

2. **MEAN**

Texture	YOLOv5 Average Mean	Faster R-CNN Average Mean
Typewriter	0.61	0.36
pager mobile	0.66	0.31

From this comparison, it's seen that YOLOv5 generally outperforms Faster R-CNN in terms of average mean precision across the different textures.

[ ]:

[ ]:

[ ]:

[ ]:

1.8 Exercise 3: Different Geometric Transformation

1.8.1 Videos is linked in google drive above

- Open Exercise_3 folder and you will find separate folder for transformations
- Open Size transform for Scaling
- Open Position transform for Translation
- Video will named as 50.mp4, 100.mp4, * .mp4 (50,100,* represents the size of the patch)
- Videos will be named as 11.mp4, 12.mp4, 21.mp4, * .mp4 (11, 12,21,* represents the position of the patch in the grid of 3x3)

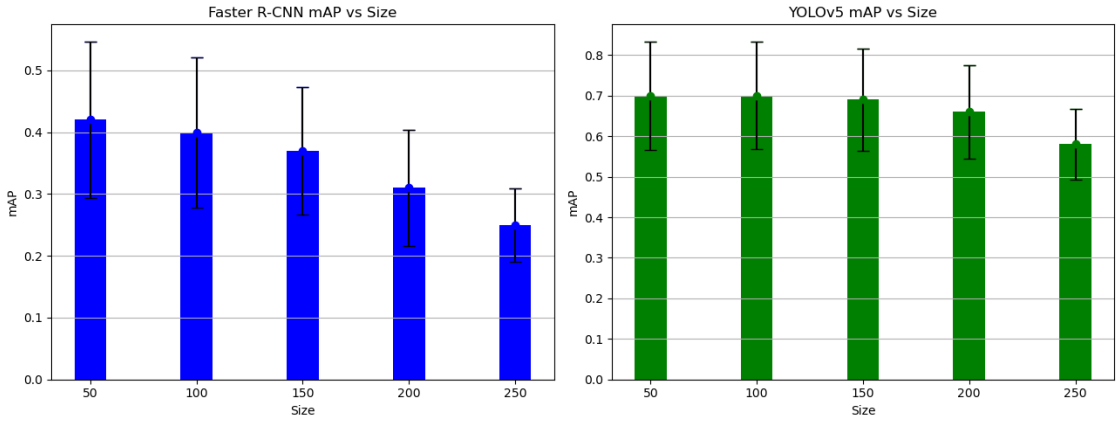
1. **Scaling**: Resizing the patch to make it larger or smaller. We used differrent size of patches placed at the centre. Sizes vary from ***50x50*** , ***100x100*** , ***150x150*** , ***200x200*** , ***250x250***

Results: Average of mAP and Std. Dev. is calculated and listed as table

Size	RCNN (mAP)	Faster R-CNN Std. Dev.	YOLO (mAP)	YOLOv5 Std. Dev.
50	0.42	0.1268	0.70	0.1338
100	0.40	0.1216	0.70	0.1328
150	0.37	0.1034	0.69	0.1255
200	0.31	0.0937	0.66	0.1154
250	0.25	0.0592	0.58	0.0881

[90]:
```
import matplotlib.pyplot as plt

# Data
sizes = [50, 100, 150, 200, 250]
rcnn_mAP = [0.42, 0.40, 0.37, 0.31, 0.25]
rcnn_std_dev = [0.1268, 0.1216, 0.1034, 0.0937, 0.0592]
yolo_mAP = [0.70, 0.70, 0.69, 0.66, 0.58]
yolo_std_dev = [0.1338, 0.1328, 0.1255, 0.1154, 0.0881]

# Plotting error bar plot for Faster R-CNN
plt.figure(figsize=(13, 5))
plt.subplot(1, 2, 1)
plt.errorbar(sizes, rcnn_mAP, yerr=rcnn_std_dev, fmt='o', color='blue',␣
 ↪capsize=5)
```

```
plt.bar(sizes, rcnn_mAP, yerr=rcnn_std_dev, color='blue', capsize=5, width=15)
plt.xlabel('Size')
plt.ylabel('mAP')
plt.title('Faster R-CNN mAP vs Size')
plt.xticks(sizes)
plt.grid(axis='y')

# Plotting error bar plot for YOLOv5
plt.subplot(1, 2, 2)
plt.errorbar(sizes, yolo_mAP, yerr=yolo_std_dev, fmt='o', color='green',␣
 ↪capsize=5)
plt.bar(sizes, yolo_mAP, yerr=yolo_std_dev, color='green', capsize=5, width=15)
plt.xlabel('Size')
plt.ylabel('mAP')
plt.title('YOLOv5 mAP vs Size')
plt.xticks(sizes)
plt.grid(axis='y')
plt.tight_layout()

# Show plots
plt.show()
```



2. **Translation**: Moving the patch horizontally and/or vertically. A grid of ***270x270*** is formed referencing the center of the plot. We use patch of size ***90x90***, which is placed in the grid which gives a 3x3 block where we place the patch in these blocks gives us 9 diffrent locations.

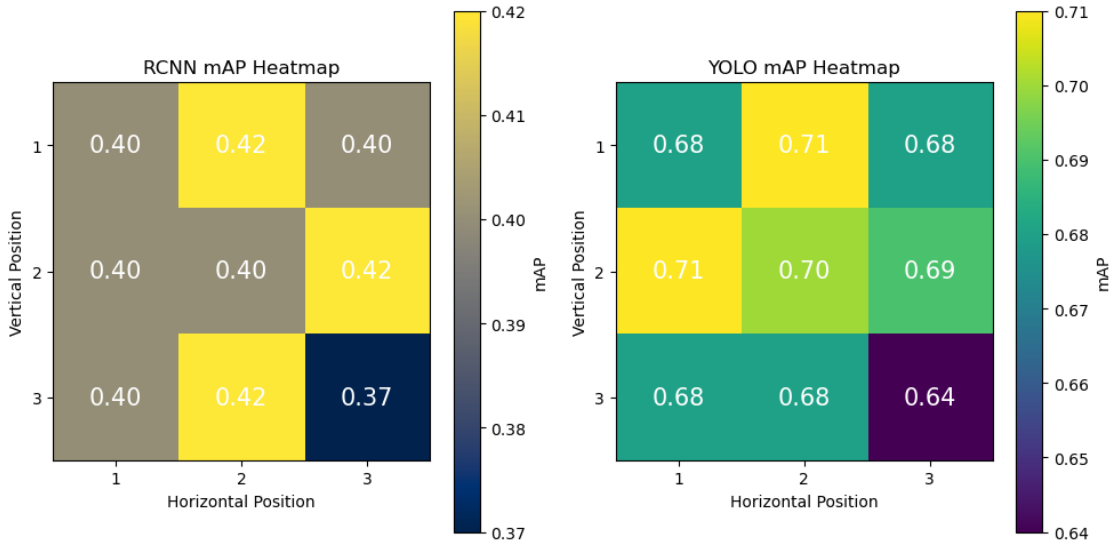
-(1,1)-	-(1,2)-	-(1,3)-
-(2,1)-	-(2,2)-	-(2,3)-
-(3,1)-	-(3,2)-	-(3,3)-

Results: Average of mAP and Std. Dev. is calculated and listed as table

Position	RCNN (mAP)	Faster R-CNN Std. Dev.	YOLO (mAP)	YOLOv5 Std. Dev.
(1,1)	0.40	0.1152	0.68	0.1251
(1,2)	0.42	0.1365	0.71	0.1133
(1,3)	0.40	0.1232	0.68	0.1226
(2,1)	0.40	0.1257	0.71	0.1281
(2,2)	0.40	0.1206	0.70	0.1280
(2,3)	0.42	0.1354	0.69	0.1255
(3,1)	0.40	0.1235	0.68	0.1337
(3,2)	0.42	0.1362	0.68	0.1297
(3,3)	0.37	0.1069	0.64	0.1457

[91]:
```
import numpy as np
import matplotlib.pyplot as plt

# Data
positions = [(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2),␣
 ↪(3, 3)]
rcnn_mAP = [0.40, 0.42, 0.40, 0.40, 0.40, 0.42, 0.40, 0.42, 0.37]
yolo_mAP = [0.68, 0.71, 0.68, 0.71, 0.70, 0.69, 0.68, 0.68, 0.64]

# Convert data to matrix form
rcnn_matrix = np.array(rcnn_mAP).reshape(3, 3)
yolo_matrix = np.array(yolo_mAP).reshape(3, 3)

# Plot heatmap for RCNN
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(rcnn_matrix, cmap='cividis', interpolation='nearest')

# Add text annotations
for i in range(3):
    for j in range(3):
        plt.text(j, i, '{:.2f}'.format(rcnn_matrix[i, j]),
                 ha='center', va='center', color='white',fontsize=15)

plt.colorbar(label='mAP')
plt.title('RCNN mAP Heatmap')
plt.xticks(np.arange(3), ['1', '2', '3'])
plt.yticks(np.arange(3), ['1', '2', '3'])
plt.xlabel('Horizontal Position')
plt.ylabel('Vertical Position')

# Plot heatmap for YOLO
plt.subplot(1, 2, 2)
plt.imshow(yolo_matrix, cmap='viridis', interpolation='nearest')
```

```
# Add text annotations
for i in range(3):
    for j in range(3):
        plt.text(j, i, '{:.2f}'.format(yolo_matrix[i, j]),
                 ha='center', va='center', color='white',fontsize=15)

plt.colorbar(label='mAP')
plt.title('YOLO mAP Heatmap')
plt.xticks(np.arange(3), ['1', '2', '3'])
plt.yticks(np.arange(3), ['1', '2', '3'])
plt.xlabel('Horizontal Position')
plt.ylabel('Vertical Position')
plt.tight_layout()

# Show plots
plt.show()
```



1.9 Heatmap of the position's impact of to the AP@0.5 metrics

1.10 RCNN Heatmap of the position's impact on the stopsign

1.11 YOLO Heatmap of the position's impact on the stopsign