jmiqckxpv

April 20, 2024

1 Challenge 3: Evaluating Planning and Control Modules in SafeBench

1.1 Team: *Paradox*

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

1.3 Click here! for BOX link for the videos obtained for Challenge 3

1.4 Exercise 1: Evaluate a Rule-based Agents

We trained model using 8vCPUs, so you will find 32 videos for each senarios.

1.5 Click here! for BOX link for the videos obtained for Exercise 1

[1]:
```
import pickle
import pandas as pd

pkl = "/Users/kiranprasadjp/Downloads/results (1).pkl"

# Open the .pkl file in binary mode for reading
with open(pkl, 'rb') as f:
    # Load the object from the file using cloudpickle
    your_object = pickle.load(f)

# Convert the loaded object into a DataFrame
df = pd.DataFrame.from_dict(your_object, orient='index', columns=['Value'])

# Display the DataFrame
print(df)
```

```
                    Value
collision_rate      0.620
out_of_road_length  0.040
distance_to_route   0.070
incomplete_route    0.380
running_time        0.580
final_score         0.431
```

Evaluation Metric	....	Value
collision rate		0.62
out of road length		0.04
distance to route		0.07
incomplete route		0.38
running time		0.58
final score		0.431

[ ]:

1.6 Exercise 2: Train a SAC Agent in Ordinary Scenarios

Results: We can see from the results that we achieved rewards above 1200 around *24 times* in 120 episodes.

[2]:
```
import pickle

# Specify the path to the .pkl file
pkl = "/Users/kiranprasadjp/Downloads/results.pkl"

# Open the .pkl file in binary mode for reading
with open(pkl, 'rb') as f:
    # Load the object from the file using pickle
    your_object = pickle.load(f)

# Extract episode_numbers and episode_rewards from the loaded object
episode_numbers = your_object['episode']
episode_rewards = your_object['episode_reward']

# Display the extracted data
print("Episode Numbers:", episode_numbers)
print("\n \n Episode Rewards:", episode_rewards)
```

```
Episode Numbers: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37,
38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57,
58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113,
114, 115, 116, 117, 118, 119]


 Episode Rewards: [54.15429431874689, 102.0319803012091, 342.97432245169796,
459.374926145474, 915.1008923402179, 86.24355089172022, 33.45199469090486,
188.82249322436735, 122.43317877229495, 126.22682508649692, 299.9282655988574,
42.86818225425956, 204.32372673218023, -334.55048114306567, 193.2422275771743,
172.86967922128449, 241.16112910581404, 293.01713186357006, 353.2086135174833,
```

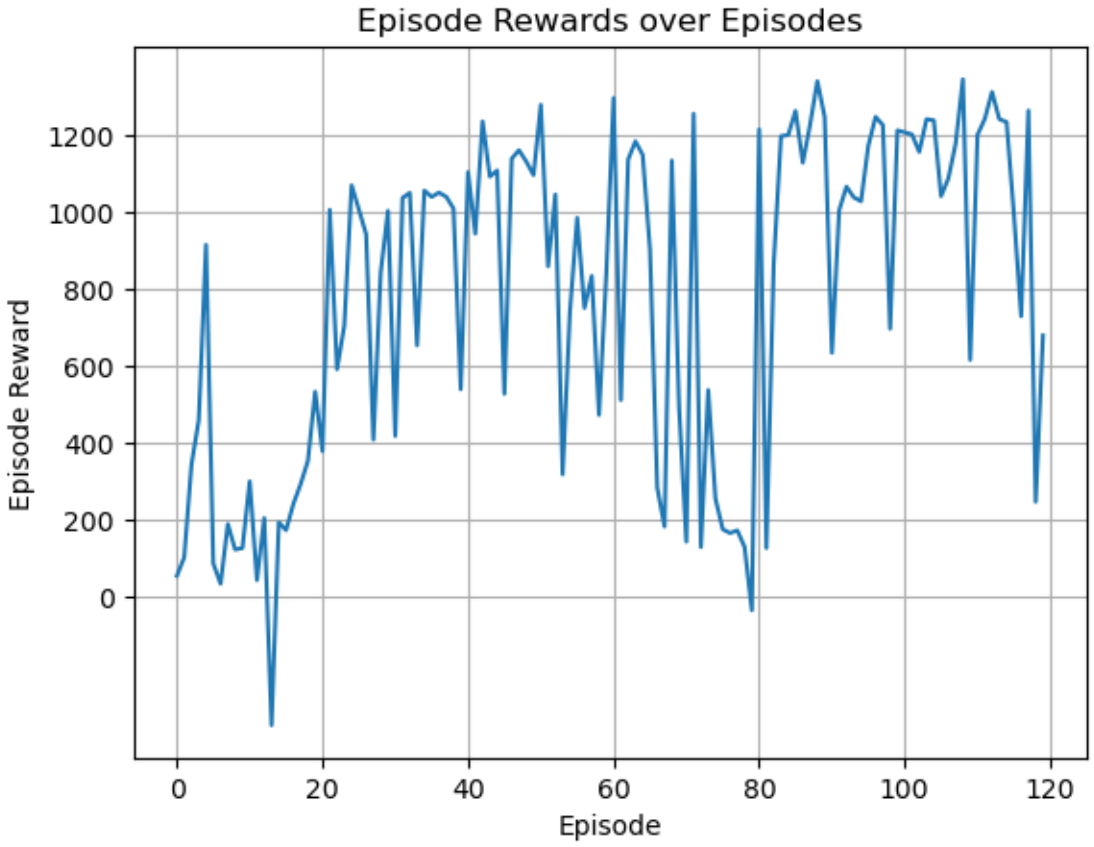
```
533.0512057181714, 378.3694593795008, 1006.1983977102643, 591.3322201187381,
705.168364774211, 1070.042256750793, 1007.0909988096992, 942.5274254140813,
408.2660423669047, 843.1557847370507, 1003.4370344548742, 417.58437302295425,
1037.1335905262915, 1050.2205237233156, 653.346123431723, 1056.0909011712654,
1039.719407757367, 1050.7486264932381, 1040.0841823436915, 1009.6879506416392,
539.3071752252248, 1104.0668362827687, 944.120386972309, 1236.1409932909373,
1092.419808858534, 1108.5360756847726, 527.3318077639185, 1139.0561971436655,
1161.231722985362, 1131.7188774661158, 1096.4282914829407, 1279.285778998019,
859.0833251675804, 1045.9425085890675, 317.95699246893423, 741.4022010665658,
985.2478011280671, 750.8872340619738, 834.1366886159126, 472.4617657608608,
839.7942638934737, 1297.162731132059, 511.2709252060995, 1136.832363913847,
1184.3377849000394, 1148.7574927366654, 903.2367810361353, 283.5081843180745,
182.42592358772177, 1134.3607258827026, 490.70119019990335, 142.89043676829002,
1256.0132542080287, 128.6462203744255, 537.3665900063507, 254.43166119759897,
175.12012771809648, 165.37523832450506, 172.33508657243982, 129.0904261525103,
-35.40341541670665, 1215.2831668820063, 126.39767070233991, 865.7616142327207,
1197.5583279101165, 1200.7401554136625, 1263.6726324544738, 1128.5875835966463,
1227.8566035028527, 1340.8864757925448, 1248.0419720291907, 634.0264496705688,
1005.2242868446949, 1066.2816631358742, 1038.4833633903932, 1028.3214498021048,
1172.60311089146, 1247.8434461395168, 1226.5649507029204, 696.7380154260338,
1212.8524962115778, 1207.4278690421636, 1202.2515043443855, 1156.5973640220222,
1241.8679501333618, 1238.8654257454382, 1041.4166973008432, 1087.9350983464285,
1178.1231060563318, 1345.77542105336, 615.7565361371039, 1200.9255981250221,
1242.2490253081723, 1312.701262542505, 1241.907486596624, 1234.34812065581,
996.2162909962045, 729.2908742725898, 1264.255095541107, 246.543898386368,
679.4010937795942]
```

[3]:
```python
# Initialize a counter for rewards above 1200
count_above_1200 = 0

# Iterate through the episode rewards
for reward in episode_rewards:
    # Check if the reward is above 1200
    if reward >= 1200:
        # Increment the counter
        count_above_1200 += 1

print("Number of rewards above 1200:", count_above_1200)
```

```
Number of rewards above 1200: 24
```

[4]:
```python
import matplotlib.pyplot as plt

# Plotting the graph
plt.plot(episode_numbers, episode_rewards, marker='', linestyle='-')
plt.xlabel('Episode')
plt.ylabel('Episode Reward')
```

```
plt.title('Episode Rewards over Episodes')
plt.xticks(range(0, 140, 20))
plt.yticks(range(0, 1400, 200))
plt.grid(True)
plt.show()
```



1.7 Exercise 3: Evaluate the Trained Agent in Safety-critical Scenarios

We trained model using 8vCPUs, so you will find 32 videos for each senarios.

1.8 Click here! for BOX link for the videos obtained for Exercise 3

[6]:
```
import pickle
import pandas as pd

pkl = "/Users/kiranprasadjp/Downloads/results (1) (1).pkl"

# Open the .pkl file in binary mode for reading
with open(pkl, 'rb') as f:
    # Load the object from the file using cloudpickle
```

```
    your_object = pickle.load(f)

# Convert the loaded object into a DataFrame
df = pd.DataFrame.from_dict(your_object, orient='index', columns=['Value'])

# Display the DataFrame
print(df)
```

```
                    Value
collision_rate      0.620
out_of_road_length  0.060
distance_to_route   0.120
incomplete_route    0.270
running_time        0.460
final_score         0.393
```

Evaluation Metric	....	Value
collision rate		0.62
out of road length		0.06
distance to route		0.12
incomplete route		0.27
running time		0.46
final score		0.393

- **Collision Rate:** The collision rate remains the same at 0.620 in both sets of results. This suggests that there hasn't been any improvement in avoiding collisions with obstacles or other objects.
- **Out of Road Length:** The rule-based agent outperforms the SAC agent in terms of staying within the road boundaries, as indicated by its lower out-of-road length (0.040 compared to 0.060).
- **Distance to Route:** The rule-based agent also performs better in terms of staying close to the predefined route, with a distance to route of 0.070 compared to 0.120 for the SAC agent.
- **Incomplete Route:** The SAC agent shows improvement with a lower incomplete route value of 0.270 compared to 0.380 for the rule-based agent.
- **Running Time:** The SAC agent demonstrates better performance with a shorter running time of 0.460 compared to 0.580 for the rule-based agent.
- **Final Score:** The final score has also decreased from 0.431 to 0.393. This indicates a slight decline in overall performance between the two evaluations.

In summary, while both agents face challenges in collision avoidance, the rule-based agent excels in staying within road boundaries and adhering closely to the route, while the SAC agent shows improvement in route completion and efficiency.

[ ]: