pa2-actor-critic-acrobot

March 29, 2023

[ ]:
```
import numpy as np
import random
import torch
import torch.nn as nn
import torch.nn.functional as F
from collections import namedtuple, deque
import torch.optim as optim
import datetime
import gym
from gym.wrappers.record_video import RecordVideo
import glob
import io
import base64
import matplotlib.pyplot as plt
from IPython.display import HTML
from pyvirtualdisplay import Display
import tensorflow as tf
from IPython import display as ipythondisplay
from PIL import Image
import tensorflow_probability as tfp
```

[ ]:
```
class ActorCriticModel(tf.keras.Model):
    """
    Defining policy and value networkss
    """
    def __init__(self,␣
↪action_size,num_layers=2,num_neurons_each_layer=[1024,512]):
        super(ActorCriticModel, self).__init__()
        self.num_layers  =num_layers
        self.neurons_each_layer = num_neurons_each_layer
        self.linears = [tf.keras.layers.Dense(self.neurons_each_layer[0],␣
↪activation=tf.nn.relu)]
        self.linears.extend([tf.keras.layers.Dense(self.neurons_each_layer[i],␣
↪activation=tf.nn.relu) for i in range(1, self.num_layers)])

        #Output Layer for policy
        self.pi_out = tf.keras.layers.Dense(action_size, activation=tf.nn.
↪softmax)
        #Output Layer for state-value
        self.v_out = tf.keras.layers.Dense(1)

    def call(self, state):
        """
        Computes policy distribution and state-value for a given state
        """
        h = None
        for i in range(self.num_layers+1):
          if i == 0:
             h = tf.nn.relu(self.linears[0](state))
          elif i < self.num_layers:
             h = tf.nn.relu(self.linears[i](h))
          else:
             return self.pi_out(h), self.v_out(h)
```

[ ]:
```
class Agent:

    def __init__(self, action_size, num_layers, num_neurons_each_layer, lr=0.
↪001, gamma=0.99, seed = 85):
        self.gamma = gamma
        self.ac_model =␣
↪ActorCriticModel(action_size=action_size,num_layers=num_layers,num_neurons_each_layer=num_n
        self.ac_model.compile(tf.keras.optimizers.Adam(learning_rate=lr))
        np.random.seed(seed)

    def sample_action(self, state):

        pi,_ = self.ac_model(state)

        action_probabilities = tfp.distributions.Categorical(probs=pi)
        sample = action_probabilities.sample()

        return int(sample.numpy()[0])

    def actor_loss(self, action, pi, delta):

        return -tf.math.log(pi[0,action]) * delta

    def critic_loss(self,delta):
        return delta**2

    @tf.function
    def learn(self, state, action, reward, next_state, done):
        with tf.GradientTape(persistent=True) as tape:
            pi, V_s = self.ac_model(state)
            _, V_s_next = self.ac_model(next_state)

            V_s = tf.squeeze(V_s)
            V_s_next = tf.squeeze(V_s_next)

            delta = reward+((self.gamma)*V_s_next) - V_s
            loss_a = self.actor_loss(action, pi, delta)
            loss_c = self.critic_loss(delta)
            loss_total = loss_a + loss_c

        gradient = tape.gradient(loss_total, self.ac_model.trainable_variables)
        self.ac_model.optimizer.apply_gradients(zip(gradient, self.ac_model.
 ↪trainable_variables))
```

[ ]:
```
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)


def Actor_critic_one_step_return(num_layers, num_neurons_each_layer,␣
 ↪LR,num_episodes,seed,TRUNCATION):
    env = gym.make('Acrobot-v1',max_episode_steps=TRUNCATION)

    #Initializing Agent
    agent = Agent( action_size=env.action_space.n, num_layers=num_layers,␣
 ↪num_neurons_each_layer=num_neurons_each_layer, lr=LR,seed = seed)
    #Number of episodes
    episodes = num_episodes
    tf.compat.v1.reset_default_graph()

    reward_list = []
    steps = []
    ep_solved = num_episodes

    begin_time = datetime.datetime.now()

    for ep in range(1, episodes + 1):
        state = env.reset().reshape(1,-1)
        done = False
        ep_rew =0
        T =0
        while not done:
            T += 1
            action = agent.sample_action(state)  # Action taken in an episode
            next_state, reward, done, info = env.step(action)  # Observing the␣
 ↪state transitions
```

```
            next_state = next_state.reshape(1,-1)
            ep_rew += reward  # Accumulate the reward
            agent.learn(state, action, reward, next_state, done) # Modify the␣
↪weights of neural network based on observed weights
            state = next_state
        reward_list.append(ep_rew)
        steps.append(T)

        if ep % 10 == 0:
            avg_rew = np.mean(reward_list[-10:])
            print('Episode ', ep, 'Reward %f' % ep_rew, 'Average Reward %f' %␣
↪avg_rew)
        if ep>1000 and (np.std(reward_list[-500:])*100/np.mean(reward_list[-500:
↪])) < 5:
            ep_solved = num_episodes
            break
        if ep % 100:
            avg_100 =  np.mean(reward_list[-100:])
            if avg_100 > -100:
                print('Stopped at Episode ',ep-100)
                ep_solved= ep-100
                break

    time_taken = datetime.datetime.now() - begin_time
    print(time_taken)
    return [reward_list,steps,ep_solved]
```

[ ]:

```
SEED = 101
def calculate_mean(array):
    lens = [len(i) for i in array]
    arr = np.ma.empty((np.max(lens),len(array)))
    arr.mask = True
    for idx, l in enumerate(array):
        arr[:len(l),idx] = l
    return arr.mean(axis = -1), arr.std(axis=-1)
```

[ ]:

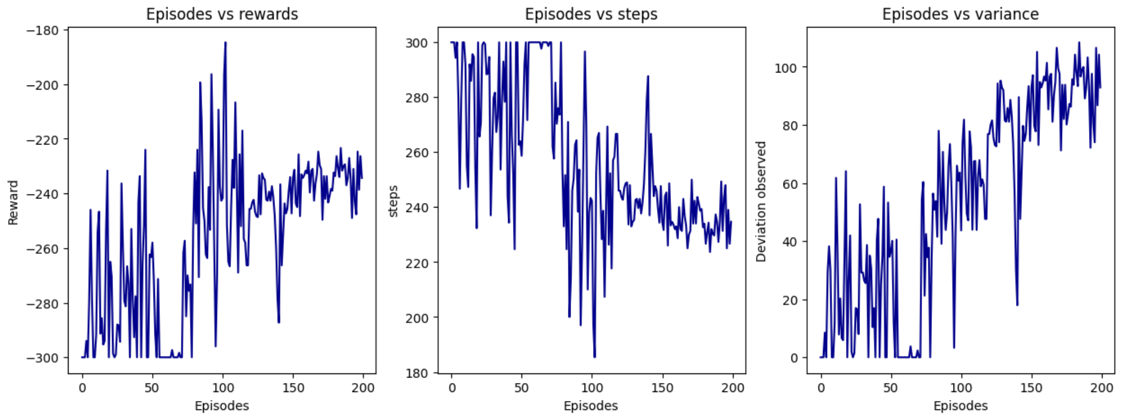
```
import numpy as np
rewards_10, steps_10,ep_10 = [],[],[]
for i in range(3):
    rewards_run, steps_run,ep = Actor_critic_one_step_return(num_layers=2,␣
↪num_neurons_each_layer=[128,128], LR=1e-4,num_episodes=500,seed=␣
↪SEED,TRUNCATION = 300)
    rewards_10.append(rewards_run)
    steps_10.append(steps_run)
    ep_10.append(ep)

reward_10_runs,error_rewards =calculate_mean(rewards_10)
```

```
steps_10_runs,error_steps = calculate_mean(steps_10)
avg_ep = np.mean(ep_10)
print(rewards_10, steps_10,ep_10)
```

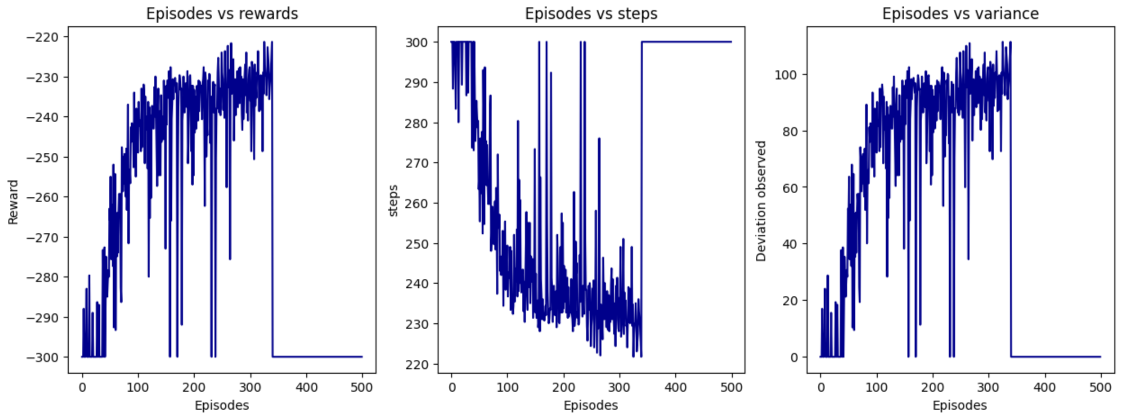
```
Episode  10 Reward -300.000000 Average Reward -289.900000
Episode  20 Reward -300.000000 Average Reward -247.000000
Episode  30 Reward -232.000000 Average Reward -282.700000
Episode  40 Reward -300.000000 Average Reward -269.000000
Episode  50 Reward -273.000000 Average Reward -253.500000
Episode  60 Reward -300.000000 Average Reward -282.500000
Episode  70 Reward -300.000000 Average Reward -299.200000
Episode  80 Reward -219.000000 Average Reward -291.900000
Episode  90 Reward -289.000000 Average Reward -232.100000
Episode  100 Reward -274.000000 Average Reward -212.600000
Episode  110 Reward -177.000000 Average Reward -231.900000
Episode  120 Reward -300.000000 Average Reward -276.600000
Episode  130 Reward -300.000000 Average Reward -300.000000
Episode  140 Reward -300.000000 Average Reward -300.000000
Episode  150 Reward -300.000000 Average Reward -300.000000
Episode  160 Reward -300.000000 Average Reward -300.000000
Episode  170 Reward -300.000000 Average Reward -300.000000
Episode  180 Reward -300.000000 Average Reward -300.000000
Episode  190 Reward -300.000000 Average Reward -300.000000
Episode  200 Reward -300.000000 Average Reward -300.000000
Episode  210 Reward -300.000000 Average Reward -300.000000
Episode  220 Reward -300.000000 Average Reward -300.000000
Episode  230 Reward -300.000000 Average Reward -300.000000
Episode  240 Reward -300.000000 Average Reward -300.000000
Episode  250 Reward -300.000000 Average Reward -300.000000
Episode  260 Reward -300.000000 Average Reward -300.000000
Episode  270 Reward -300.000000 Average Reward -300.000000
Episode  280 Reward -300.000000 Average Reward -300.000000
Episode  290 Reward -300.000000 Average Reward -300.000000
Episode  300 Reward -300.000000 Average Reward -300.000000
Episode  310 Reward -300.000000 Average Reward -300.000000
Episode  320 Reward -300.000000 Average Reward -300.000000
Episode  330 Reward -300.000000 Average Reward -300.000000
Episode  340 Reward -300.000000 Average Reward -300.000000
Episode  350 Reward -300.000000 Average Reward -300.000000
Episode  360 Reward -300.000000 Average Reward -300.000000
Episode  370 Reward -300.000000 Average Reward -300.000000
Episode  380 Reward -300.000000 Average Reward -300.000000
Episode  390 Reward -300.000000 Average Reward -300.000000
Episode  400 Reward -300.000000 Average Reward -300.000000
Episode  410 Reward -300.000000 Average Reward -300.000000
Episode  420 Reward -300.000000 Average Reward -300.000000
Episode  430 Reward -300.000000 Average Reward -300.000000
Episode  440 Reward -300.000000 Average Reward -300.000000
Episode  450 Reward -300.000000 Average Reward -300.000000
Episode  460 Reward -300.000000 Average Reward -300.000000
Episode  470 Reward -300.000000 Average Reward -300.000000
Episode  480 Reward -300.000000 Average Reward -300.000000
Episode  490 Reward -300.000000 Average Reward -300.000000
Episode  500 Reward -300.000000 Average Reward -300.000000
0:12:00.709434
Episode  10 Reward -300.000000 Average Reward -300.000000
Episode  20 Reward -300.000000 Average Reward -300.000000
Episode  30 Reward -300.000000 Average Reward -300.000000
Episode  40 Reward -300.000000 Average Reward -300.000000
Episode  50 Reward -300.000000 Average Reward -300.000000
Episode  60 Reward -300.000000 Average Reward -300.000000
Episode  70 Reward -300.000000 Average Reward -300.000000
Episode  80 Reward -300.000000 Average Reward -300.000000
Episode  90 Reward -300.000000 Average Reward -300.000000
Episode  100 Reward -300.000000 Average Reward -300.000000
Episode  110 Reward -300.000000 Average Reward -300.000000
Episode  120 Reward -300.000000 Average Reward -300.000000
Episode  130 Reward -300.000000 Average Reward -300.000000
Episode  140 Reward -300.000000 Average Reward -300.000000
Episode  150 Reward -300.000000 Average Reward -300.000000
Episode  160 Reward -300.000000 Average Reward -300.000000
Episode  170 Reward -300.000000 Average Reward -300.000000
Episode  180 Reward -300.000000 Average Reward -300.000000
Episode  190 Reward -300.000000 Average Reward -300.000000
Episode  200 Reward -300.000000 Average Reward -300.000000
Episode  210 Reward -300.000000 Average Reward -300.000000
Episode  220 Reward -300.000000 Average Reward -300.000000
Episode  230 Reward -300.000000 Average Reward -300.000000
Episode  240 Reward -300.000000 Average Reward -300.000000
Episode  250 Reward -300.000000 Average Reward -300.000000
Episode  260 Reward -300.000000 Average Reward -300.000000
Episode  270 Reward -300.000000 Average Reward -300.000000
Episode  280 Reward -300.000000 Average Reward -300.000000
Episode  290 Reward -300.000000 Average Reward -300.000000
Episode  300 Reward -300.000000 Average Reward -300.000000
Episode  310 Reward -300.000000 Average Reward -300.000000
Episode  320 Reward -300.000000 Average Reward -300.000000
Episode  330 Reward -300.000000 Average Reward -300.000000
Episode  340 Reward -300.000000 Average Reward -300.000000
Episode  350 Reward -300.000000 Average Reward -300.000000
Episode  360 Reward -300.000000 Average Reward -300.000000
Episode  370 Reward -300.000000 Average Reward -300.000000
Episode  380 Reward -300.000000 Average Reward -300.000000
Episode  390 Reward -300.000000 Average Reward -300.000000
Episode  400 Reward -300.000000 Average Reward -300.000000
Episode  410 Reward -300.000000 Average Reward -300.000000
Episode  420 Reward -300.000000 Average Reward -300.000000
Episode  430 Reward -300.000000 Average Reward -300.000000
Episode  440 Reward -300.000000 Average Reward -300.000000
Episode  450 Reward -300.000000 Average Reward -300.000000
Episode  460 Reward -300.000000 Average Reward -300.000000
Episode  470 Reward -300.000000 Average Reward -300.000000
Episode  480 Reward -300.000000 Average Reward -300.000000
Episode  490 Reward -300.000000 Average Reward -300.000000
Episode  500 Reward -300.000000 Average Reward -300.000000
0:12:03.668842
Episode  10 Reward -300.000000 Average Reward -279.500000
Episode  20 Reward -300.000000 Average Reward -276.300000
Episode  30 Reward -247.000000 Average Reward -257.200000
Episode  40 Reward -300.000000 Average Reward -271.500000
Episode  50 Reward -217.000000 Average Reward -239.700000
Episode  60 Reward -300.000000 Average Reward -285.300000
Episode  70 Reward -295.000000 Average Reward -299.500000
Episode  80 Reward -241.000000 Average Reward -241.000000
Episode  90 Reward -202.000000 Average Reward -191.500000
Episode  100 Reward -154.000000 Average Reward -216.100000
Episode  110 Reward -143.000000 Average Reward -165.200000
Episode  120 Reward -137.000000 Average Reward -171.400000
Episode  130 Reward -103.000000 Average Reward -126.900000
Episode  140 Reward -236.000000 Average Reward -140.200000
Episode  150 Reward -142.000000 Average Reward -150.600000
Episode  160 Reward -95.000000 Average Reward -108.200000
Episode  170 Reward -89.000000 Average Reward -98.900000
Episode  180 Reward -97.000000 Average Reward -115.200000
Episode  190 Reward -103.000000 Average Reward -93.100000
Episode  200 Reward -103.000000 Average Reward -106.800000
Episode  210 Reward -95.000000 Average Reward -105.600000
Episode  220 Reward -73.000000 Average Reward -88.700000
Episode  230 Reward -80.000000 Average Reward -122.900000
Episode  240 Reward -77.000000 Average Reward -110.400000
Episode  250 Reward -94.000000 Average Reward -90.800000
Episode  260 Reward -89.000000 Average Reward -94.800000
Episode  270 Reward -125.000000 Average Reward -90.600000
Episode  280 Reward -77.000000 Average Reward -103.100000
Episode  290 Reward -98.000000 Average Reward -96.600000
Stopped at Episode  195
0:03:59.341325
[[-300.0, -300.0, -300.0, -282.0, -300.0, -300.0, -217.0, -300.0, -300.0,
-300.0, -275.0, -167.0, -226.0, -281.0, -257.0, -286.0, -286.0, -246.0, -146.0,
-300.0, -260.0, -300.0, -300.0, -300.0, -297.0, -300.0, -300.0, -300.0, -238.0,
-232.0, -300.0, -300.0, -262.0, -218.0, -300.0, -216.0, -235.0, -300.0, -259.0,
-300.0, -218.0, -211.0, -300.0, -241.0, -235.0, -157.0, -300.0, -300.0, -300.0,
-273.0, -211.0, -300.0, -300.0, -300.0, -214.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -292.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-219.0, -235.0, -272.0, -184.0, -300.0, -188.0, -178.0, -225.0, -150.0, -300.0,
-289.0, -235.0, -160.0, -149.0, -222.0, -280.0, -296.0, -210.0, -145.0, -155.0,
-274.0, -195.0, -157.0, -135.0, -291.0, -300.0, -300.0, -300.0, -254.0, -210.0,
-177.0, -267.0, -300.0, -229.0, -300.0, -170.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0], [-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0], [-300.0, -300.0, -300.0, -300.0, -300.0, -238.0, -221.0,
-236.0, -300.0, -300.0, -300.0, -296.0, -214.0, -293.0, -300.0, -300.0, -296.0,
-215.0, -249.0, -300.0, -235.0, -211.0, -296.0, -300.0, -300.0, -264.0, -265.0,
-283.0, -171.0, -247.0, -238.0, -244.0, -238.0, -300.0, -300.0, -243.0, -300.0,
-278.0, -274.0, -300.0, -212.0, -190.0, -300.0, -256.0, -220.0, -215.0, -300.0,
-300.0, -187.0, -217.0, -263.0, -215.0, -275.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0, -300.0,
-300.0, -300.0, -295.0, -300.0, -300.0, -185.0, -172.0, -255.0, -210.0, -227.0,
-220.0, -300.0, -241.0, -162.0, -181.0, -188.0, -212.0, -110.0, -162.0, -211.0,
-300.0, -187.0, -202.0, -178.0, -300.0, -140.0, -153.0, -204.0, -292.0, -300.0,
-183.0, -257.0, -154.0, -230.0, -133.0, -119.0, -164.0, -195.0, -200.0, -135.0,
-129.0, -204.0, -143.0, -143.0, -207.0, -148.0, -156.0, -181.0, -170.0, -174.0,
-199.0, -199.0, -137.0, -137.0, -130.0, -127.0, -140.0, -145.0, -146.0, -100.0,
-143.0, -98.0, -103.0, -105.0, -127.0, -128.0, -118.0, -128.0, -112.0, -124.0,
-144.0, -180.0, -236.0, -262.0, -110.0, -199.0, -164.0, -131.0, -142.0, -137.0,
-117.0, -102.0, -142.0, -103.0, -94.0, -131.0, -135.0, -77.0, -145.0, -99.0,
-103.0, -100.0, -95.0, -98.0, -85.0, -119.0, -95.0, -93.0, -128.0, -109.0,
-99.0, -74.0, -89.0, -93.0, -149.0, -101.0, -126.0, -101.0, -130.0, -123.0,
-115.0, -117.0, -97.0, -101.0, -79.0, -92.0, -102.0, -70.0, -95.0, -90.0, -88.0,
-111.0, -103.0, -81.0, -103.0, -147.0, -93.0, -129.0, -143.0, -74.0, -116.0,
-79.0, -103.0, -115.0, -116.0, -99.0, -159.0, -110.0, -83.0, -84.0, -102.0,
-93.0, -95.0, -124.0, -83.0, -75.0, -106.0, -90.0, -81.0, -87.0, -87.0, -81.0,
-73.0, -99.0, -100.0, -193.0, -150.0, -92.0, -191.0, -67.0, -90.0, -167.0,
-80.0, -85.0, -245.0, -112.0, -79.0, -81.0, -80.0, -85.0, -100.0, -160.0, -77.0,
-92.0, -73.0, -79.0, -98.0, -90.0, -105.0, -113.0, -100.0, -64.0, -94.0, -89.0,
-90.0, -80.0, -110.0, -82.0, -85.0, -98.0, -100.0, -125.0, -89.0, -83.0, -65.0,
-106.0, -91.0, -103.0, -91.0, -83.0, -79.0, -80.0, -125.0, -92.0, -77.0, -101.0,
-108.0, -136.0, -120.0, -104.0, -107.0, -109.0, -77.0, -108.0, -85.0, -173.0,
-87.0, -101.0, -85.0, -78.0, -77.0, -74.0, -98.0, -79.0, -84.0, -80.0, -79.0,
-91.0]] [[300, 300, 300, 283, 300, 300, 218, 300, 300, 300, 276, 168, 227, 282,
258, 287, 287, 247, 147, 300, 261, 300, 300, 300, 298, 300, 300, 300, 239, 233,
300, 300, 263, 219, 300, 217, 236, 300, 260, 300, 219, 212, 300, 242, 236, 158,
300, 300, 300, 274, 212, 300, 300, 300, 215, 300, 300, 300, 300, 300, 300, 300,
300, 300, 293, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 220, 236, 273, 185, 300, 189, 179, 226, 151, 300, 290, 236, 161, 150, 223,
281, 297, 211, 146, 156, 275, 196, 158, 136, 292, 300, 300, 300, 255, 211, 178,
268, 300, 230, 300, 171, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300], [300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300], [300, 300, 300, 300,
300, 239, 222, 237, 300, 300, 300, 297, 215, 294, 300, 300, 297, 216, 250, 300,
236, 212, 297, 300, 300, 265, 266, 284, 172, 248, 239, 245, 239, 300, 300, 244,
300, 279, 275, 300, 213, 191, 300, 257, 221, 216, 300, 300, 188, 218, 264, 216,
276, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300, 300,
300, 296, 300, 300, 186, 173, 256, 211, 228, 221, 300, 242, 163, 182, 189, 213,
111, 163, 212, 300, 188, 203, 179, 300, 141, 154, 205, 293, 300, 184, 258, 155,
231, 134, 120, 165, 196, 201, 136, 130, 205, 144, 144, 208, 149, 157, 182, 171,
175, 200, 200, 138, 138, 131, 128, 141, 146, 147, 101, 144, 99, 104, 106, 128,
129, 119, 129, 113, 125, 145, 181, 237, 263, 111, 200, 165, 132, 143, 138, 118,
103, 143, 104, 95, 132, 136, 78, 146, 100, 104, 101, 96, 99, 86, 120, 96, 94,
129, 110, 100, 75, 90, 94, 150, 102, 127, 102, 131, 124, 116, 118, 98, 102, 80,
93, 103, 71, 96, 91, 89, 112, 104, 82, 104, 148, 94, 130, 144, 75, 117, 80, 104,
116, 117, 100, 160, 111, 84, 85, 103, 94, 96, 125, 84, 76, 107, 91, 82, 88, 88,
82, 74, 100, 101, 194, 151, 93, 192, 68, 91, 168, 81, 86, 246, 113, 80, 82, 81,
86, 101, 161, 78, 93, 74, 80, 99, 91, 106, 114, 101, 65, 95, 90, 91, 81, 111,
83, 86, 99, 101, 126, 90, 84, 66, 107, 92, 104, 92, 84, 80, 81, 126, 93, 78,
102, 109, 137, 121, 105, 108, 110, 78, 109, 86, 174, 88, 102, 86, 79, 78, 75,
99, 80, 85, 81, 80, 92]] [500, 500, 195]
```

[ ]:
```
reward_10_runs,error_rewards =calculate_mean(rewards_10)
steps_10_runs,error_steps = calculate_mean(steps_10)
```

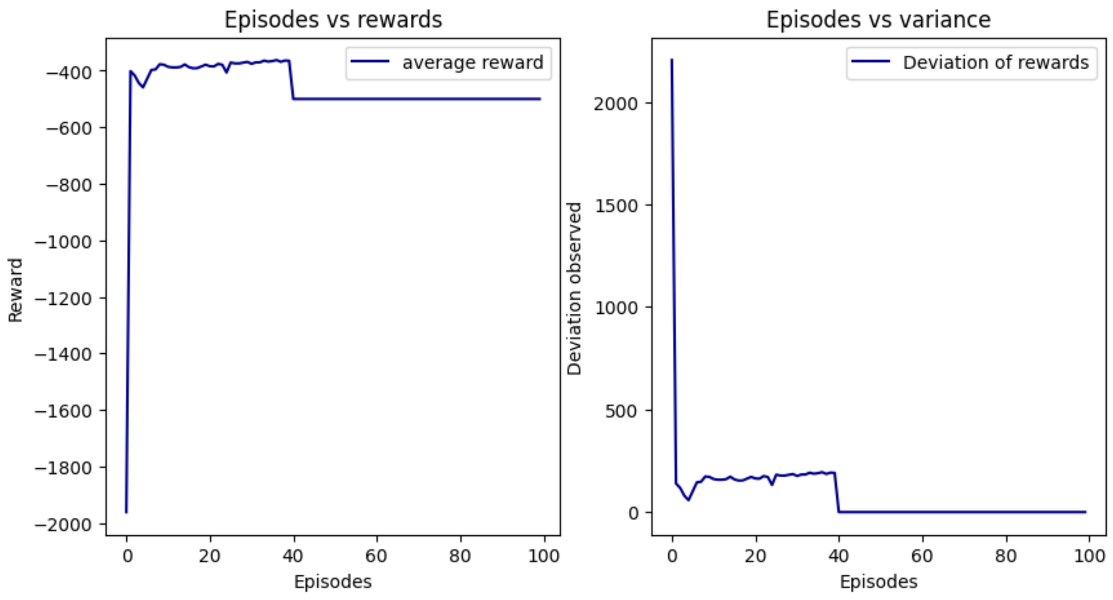
[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
plt.figure(figsize = (15,5))
plt.subplot(131)
plt.plot(np.arange(len(reward_10_runs[:-300])), reward_10_runs[:-300], color =␣
  ↪'darkblue')
plt.title("Episodes vs rewards")
plt.xlabel("Episodes")
plt.ylabel('Reward')
plt.subplot(132)
plt.plot(np.arange(len(steps_10_runs[:-300])), steps_10_runs[:-300], color =␣
  ↪'darkblue')
plt.title("Episodes vs steps")
plt.xlabel("Episodes")
plt.ylabel('steps')
plt.subplot(133)
plt.plot(np.arange(len(error_rewards[:-300])), error_rewards[:-300], color =␣
  ↪'darkblue')
plt.title("Episodes vs variance")
plt.xlabel("Episodes")
plt.ylabel('Deviation observed')
plt.show()
```



```
[ ]: import matplotlib.pyplot as plt
     import numpy as np
     plt.figure(figsize = (15,5))
     plt.subplot(131)
     plt.plot(np.arange(len(reward_10_runs)), reward_10_runs, color = 'darkblue')
     plt.title("Episodes vs rewards")
     plt.xlabel("Episodes")
     plt.ylabel('Reward')
     plt.subplot(132)
     plt.plot(np.arange(len(steps_10_runs)), steps_10_runs, color = 'darkblue')
     plt.title("Episodes vs steps")
     plt.xlabel("Episodes")
     plt.ylabel('steps')
     plt.subplot(133)
     plt.plot(np.arange(len(error_rewards)), error_rewards, color = 'darkblue')
     plt.title("Episodes vs variance")
     plt.xlabel("Episodes")
     plt.ylabel('Deviation observed')
     plt.show()
```



[ ]:
```
reward = [-308.8,
-204.6,
-250.6,
-330.7,
-377.,0
-284.3,
-192.3,
-186.3,
-131.6,
  -136.5,
  -158.6,
  -165.,0
  -165.1,
  -159.8,
  -133.8,
  -162.3,
  -173.9,
  -172.5,
-154.2,
  -136.1,
  -152.3,
  -154.5,
  -127.5,
  -137.1,
  -219.5,
  -113.4,
  -122.1,
  -122.7,
  -114.5,
  -106.8,
  -126.5,
-111.7,
-111.6,
  -92.9,
  -103.,0
-97.5,
-87.1,
  -105.6,
-91.9,
  -94.9]

reward_2 = [-256.000000
-362.200000
-500.000000
-376.100000
-455.100000
-456.100000
-344.200000
-260.300000
  -243.400000
  -240.500000
  -245.800000
  -409.900000
  -435.500000
  -490.500000]
counter = 160
while counter<=1000:
  reward_2.append(-500)
  counter+=10


reward_3 = []
co = 10
while(co <= 1000):
  reward_3.append(-500)
  co += 10


reward = [reward,reward_2, reward_3]
```



[ ]:
```
import wandb
!wandb login --relogin
```

```
import pprint
import numpy as np

sweep_config = {
    'method': 'bayes'
}

metric = {
    'name' : 'num_episodes_tosolve_cartpole_full_ac',
    'goal' : 'minimize'
}
sweep_config['metric'] = metric

parameters_dict = {
    'NUM_NEURONS_EACH_LAYER' : {
        'values': [64,90,128]
      },
    'NUM_LAYERS' : {
        'values': [2,4,8]
    },
    'NUM_EPISODES': {
      'values': [1800,2000,2500]
    }

    }
sweep_config['parameters'] = parameters_dict

import math
parameters_dict.update({
    'LR': {

        'distribution': 'uniform',
        'min': (5e-5),
        'max': (5e-4)
      }

    })

pprint.pprint(sweep_config)

sweep_id = wandb.sweep(sweep_config, project="Hyper parameter tuning RL full␣
 ↪return - v1")


def train(config = None):
  with wandb.init(config = config):
```

```
  config = wandb.config
  LR = config.LR                ## learning rate
  NUM_NEURONS_EACH_LAYER = config.NUM_NEURONS_EACH_LAYER  ##number of neurons␣
↪in each hidden layer
  NUM_LAYERS =    config.NUM_LAYERS              ##number of layers for the␣
↪neural network
  NUM_EPISODES = config.NUM_EPISODES
  steps_over_10_runs  = []
  rewards_over_10_runs = []
  episodes = []
  min_len = np.inf
  for run in range(3):
    print("Run: ",run+1,"␣
↪------------------------------------------------------------")
    env = gym.make('CartPole-v1')

    seed = np.random.randint(low=0, high = 100)
    env.seed(seed)
    state_shape = env.observation_space.shape[0]
    action_shape = env.action_space.n

    begin_time = datetime.datetime.now()   #(num_layers,␣
↪num_neurons_each_layer, LR,num_episodes,seed):
    rewards_each_ep, steps_each_ep, episodes_solved =␣
↪Actor_critic_one_step_return(NUM_LAYERS,␣
↪[NUM_NEURONS_EACH_LAYER]*NUM_LAYERS, LR,NUM_EPISODES,seed)
    steps_over_10_runs.append(steps_each_ep)
    rewards_over_10_runs.append(rewards_each_ep)
    episodes.append(episodes_solved)

    if len(rewards_each_ep) < min_len:
      min_len = len(rewards_each_ep)
    time_taken = datetime.datetime.now() - begin_time

    print(time_taken)

  rewards = np.array(rewards_over_10_runs)
  avg_episodes = int(np.array(episodes).mean())
  steps = np.array(steps_over_10_runs)

  y, error_reward = calculate_mean(rewards[:avg_episodes])
  plt.figure(figsize = (15,5))
  plt.subplot(121)
 # for i in range(len(rewards_over_10_runs)):
 #   plt.plot(np.
↪arange(len(rewards_over_10_runs[i])),rewards_over_10_runs[i],label='run␣
↪'+str(i))
```

```
    plt.plot(np.arange(len(y))+1, y, color='darkblue',label='average')
    plt.title("Episodes vs rewards")
    plt.xlabel("Number of episodes")
    plt.ylabel("Reward per episode")
    plt.legend()
    plt.subplot(122)
    y, error = calculate_mean(steps[:avg_episodes])
    plt.plot(np.arange(len(y))+1, y, color='darkblue',label='average')
    plt.title("Episodes vs steps ")
    plt.xlabel("Number of episodes")
    plt.ylabel("Steps taken per episode")
    plt.legend()
    plt.subplot(133)
    plt.plot(np.arange(len(error_reward))+1, error_reward ,␣
↪color='darkblue',label= 'Deviation of rewards') #plotting the variance = std␣
↪squared
    plt.title("Episodes vs Variance ")
    plt.xlabel("Number of episodes")
    plt.ylabel("Observed Variance")
    plt.legend()
    plt.savefig('hyp_cartpole_ac_one_step_return'+str(NUM_EPISODES)+str(LR)+'.
↪jpg', dpi = 250)
    plt.show()


    for i in range(avg_episodes):
      print("Episode: ",i+1, 'Number of steps: ',round(y[i]))
    wandb.log({"num_episodes_tosolve_cartpole_full_ac":avg_episodes})
```

```
wandb: ERROR Find detailed error logs at: c:\Users\CFI
Workstation\Documents\RL\wandb\debug-cli.CFI Workstation.log
Error: api_key not configured (no-tty). call wandb login [your_api_key]
```

```
{'method': 'bayes',
 'metric': {'goal': 'minimize',
            'name': 'num_episodes_tosolve_cartpole_full_ac'},
 'parameters': {'LR': {'distribution': 'uniform', 'max': 0.0005, 'min': 5e-05},
                'NUM_EPISODES': {'values': [1800, 2000, 2500]},
                'NUM_LAYERS': {'values': [2, 4, 8]},
                'NUM_NEURONS_EACH_LAYER': {'values': [64, 90, 128]}}}
```

```
C:\Users\CFI Workstation\AppData\Local\Packages\PythonSoftwareFoundation.Python.
3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-
packages\notebook\utils.py:280: DeprecationWarning: distutils Version classes
are deprecated. Use packaging.version instead.
  return LooseVersion(v) >= LooseVersion(check)
Failed to detect the name of this notebook, you can set it manually with the
WANDB_NOTEBOOK_NAME environment variable to enable code saving.
```

```
Create sweep with ID: ofvxspxo
Sweep URL: https://wandb.ai/me19b190/Hyper%20parameter%20tuning%20RL%20full%20re
turn%20-%20v1/sweeps/ofvxspxo
```

```
[ ]: wandb.agent(sweep_id, train, count=3)
```

```
wandb: Agent Starting Run: 0g4dbjhd with config:
wandb:     LR: 0.0004888636786360583
wandb:     NUM_EPISODES: 2000
wandb:     NUM_LAYERS: 8
wandb:     NUM_NEURONS_EACH_LAYER: 128
Failed to detect the name of this notebook, you can set it manually with the
WANDB_NOTEBOOK_NAME environment variable to enable code saving.
wandb: Currently logged in as: me19b190. Use `wandb

login --relogin` to force relogin
C:\Users\CFI Workstation\AppData\Local\Packages\PythonSoftwareFoundation.Python.
3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-
packages\wandb\sdk\lib\ipython.py:47: DeprecationWarning: Importing display from
IPython.core.display is deprecated since IPython 7.14, please import from
IPython display
  from IPython.core.display import HTML, display  # type: ignore

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

C:\Users\CFI Workstation\AppData\Local\Packages\PythonSoftwareFoundation.Python.
3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-
packages\gym\core.py:317: DeprecationWarning: WARN: Initializing wrapper in

old step API which returns one bool instead of two. It is recommended to set

`new_step_api=True` to use new step API. This will be the default behaviour in

future.
  deprecation(
C:\Users\CFI Workstation\AppData\Local\Packages\PythonSoftwareFoundation.Python.
3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-
packages\gym\wrappers\step_api_compatibility.py:39: DeprecationWarning:
WARN: Initializing environment in old step API which returns one bool

instead of two. It is recommended to set `new_step_api=True` to use new step

API. This will be the default behaviour in future.
  deprecation(
C:\Users\CFI Workstation\AppData\Local\Packages\PythonSoftwareFoundation.Python.
3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-
packages\gym\core.py:256: DeprecationWarning: WARN: Function
`env.seed(seed)` is marked as deprecated and will be removed in the future.
Please use `env.reset(seed=seed)` instead.
  deprecation(
Run:  1  ------------------------------------------------------------
Episode  10 Reward -500.000000 Average Reward -500.000000
Episode  20 Reward -500.000000 Average Reward -500.000000
Episode  30 Reward -500.000000 Average Reward -500.000000
Episode  40 Reward -500.000000 Average Reward -500.000000
Episode  50 Reward -500.000000 Average Reward -500.000000
```

```
[ ]: #execution interrupted due to bad hyperparameter setting

steps_over_10_runs  = []
rewards_over_10_runs = []
episodes = []
min_len = np.inf
for run in range(1):
    print("Run: ",run+1,"␣
  ↪------------------------------------------------------------")
    env = gym.make('CartPole-v1')

    seed = np.random.randint(low=0, high = 100)
    env.seed(seed)
    state_shape = env.observation_space.shape[0]
    action_shape = env.action_space.n

    begin_time = datetime.datetime.now()   #(num_layers,␣
  ↪num_neurons_each_layer, LR,num_episodes,seed):
    rewards_each_ep, steps_each_ep, episodes_solved =␣
  ↪Actor_critic_one_step_return(NUM_LAYERS, NUM_NEURONS_EACH_LAYER,␣
  ↪LR,NUM_EPISODES,seed)
    steps_over_10_runs.append(steps_each_ep)
    rewards_over_10_runs.append(rewards_each_ep)
    episodes.append(episodes_solved)

    if len(rewards_each_ep) < min_len:
        min_len = len(rewards_each_ep)
    time_taken = datetime.datetime.now() - begin_time

    print(time_taken)

rewards = np.array(rewards_over_10_runs)
```

```
avg_episodes = int(np.array(episodes).mean())
steps = np.array(steps_over_10_runs)

y, error_reward = calculate_mean(rewards[:avg_episodes])
plt.figure(figsize = (15,5))
plt.subplot(121)
   # for i in range(len(rewards_over_10_runs)):
   #   plt.plot(np.
 ↪arange(len(rewards_over_10_runs[i])),rewards_over_10_runs[i],label='run␣
 ↪'+str(i))
plt.plot(np.arange(len(y))+1, y, color='darkblue',label='average')
plt.title("Episodes vs rewards")
plt.xlabel("Number of episodes")
plt.ylabel("Reward per episode")
plt.legend()
plt.subplot(122)
y, error = calculate_mean(steps[:avg_episodes])
plt.plot(np.arange(len(y))+1, y, color='darkblue',label='average')
plt.title("Episodes vs steps ")
plt.xlabel("Number of episodes")
plt.ylabel("Steps taken per episode")
plt.legend()
plt.subplot(133)
plt.plot(np.arange(len(error_reward))+1, error_reward , color='darkblue',label=␣
 ↪'Deviation of rewards') #plotting the variance = std squared
plt.title("Episodes vs Variance ")
plt.xlabel("Number of episodes")
plt.ylabel("Observed Variance")
plt.legend()
plt.savefig('hyp_cartpole_ac_one_step_return'+str(NUM_EPISODES)+str(LR)+'.jpg',␣
 ↪dpi = 250)
plt.show()


for i in range(avg_episodes):
    print("Episode: ",i+1, 'Number of steps: ',round(y[i]))
```

```
Run:  1 ------------------------------------------------------------
Episode  10 Reward -205.000000 Average Reward -252.300000
Episode  20 Reward -141.000000 Average Reward -193.500000
Episode  30 Reward -220.000000 Average Reward -173.200000
Episode  40 Reward -163.000000 Average Reward -167.200000
Episode  50 Reward -229.000000 Average Reward -150.100000
Episode  60 Reward -169.000000 Average Reward -158.100000
Episode  70 Reward -140.000000 Average Reward -158.200000
Episode  80 Reward -232.000000 Average Reward -133.200000
Episode  90 Reward -500.000000 Average Reward -316.100000
Episode  100 Reward -500.000000 Average Reward -500.000000
Episode  110 Reward -500.000000 Average Reward -500.000000
Episode  120 Reward -500.000000 Average Reward -500.000000
Episode  130 Reward -500.000000 Average Reward -500.000000
Episode  140 Reward -500.000000 Average Reward -500.000000
Episode  150 Reward -500.000000 Average Reward -500.000000
Episode  160 Reward -500.000000 Average Reward -500.000000
Episode  170 Reward -500.000000 Average Reward -500.000000
Episode  180 Reward -500.000000 Average Reward -500.000000
Episode  190 Reward -500.000000 Average Reward -500.000000
Episode  200 Reward -500.000000 Average Reward -500.000000
Episode  210 Reward -500.000000 Average Reward -500.000000
Episode  220 Reward -500.000000 Average Reward -500.000000
Episode  230 Reward -500.000000 Average Reward -500.000000
Episode  240 Reward -500.000000 Average Reward -500.000000
Episode  250 Reward -500.000000 Average Reward -500.000000
Episode  260 Reward -500.000000 Average Reward -500.000000
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[11], line 15
     12 action_shape = env.action_space.n
     14 begin_time = datetime.datetime.now()   #(num_layers,␣
 ↪num_neurons_each_layer, LR,num_episodes,seed):
---> 15 rewards_each_ep, steps_each_ep, episodes_solved =␣
 ↪Actor_critic_one_step_return(NUM_LAYERS, NUM_NEURONS_EACH_LAYER,␣
 ↪LR,NUM_EPISODES,seed)
     16 steps_over_10_runs.append(steps_each_ep)
     17 rewards_over_10_runs.append(rewards_each_ep)

Cell In[6], line 31, in Actor_critic_one_step_return(num_layers,␣
 ↪num_neurons_each_layer, LR, num_episodes, seed)
     29     next_state = next_state.reshape(1,-1)
     30     ep_rew += reward  ##Updating episode reward
---> 31     agent.learn(state, action, reward, next_state, done) ##Update␣
 ↪Parameters
     32     state = next_state ##Updating State
     33 reward_list.append(ep_rew)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
 ↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\tensorflow\python\util\t
 ↪py:150, in filter_traceback.<locals>.error_handler(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)
```

```
File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
 ↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\tensorflow\python\eager
 ↪py:894, in Function.__call__(self, *args, **kwds)
    891 compiler = "xla" if self._jit_compile else "nonXla"
    893 with OptionalXlaContext(self._jit_compile):
--> 894   result = self._call(*args, **kwds)
    896 new_tracing_count = self.experimental_get_tracing_count()
    897 without_tracing = (tracing_count == new_tracing_count)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
 ↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\tensorflow\python\eager
 ↪py:926, in Function._call(self, *args, **kwds)
    923   self._lock.release()
    924   # In this case we have created variables on the first call, so we run
 ↪the
    925   # defunned version which is guaranteed to never create variables.
--> 926   return self._no_variable_creation_fn(*args, **kwds)  # pylint:␣
 ↪disable=not-callable
    927 elif self._variable_creation_fn is not None:
    928   # Release the lock early so that multiple threads can perform the cal
    929   # in parallel.
    930   self._lock.release()

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
 ↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\tensorflow\python\eager
 ↪py:143, in TracingCompiler.__call__(self, *args, **kwargs)
    140 with self._lock:
    141   (concrete_function,
    142    filtered_flat_args) = self._maybe_define_function(args, kwargs)
--> 143 return concrete_function._call_flat(
    144     filtered_flat_args, captured_inputs=concrete_function.
 ↪captured_inputs)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
 ↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\tensorflow\python\eager
 ↪py:1757, in ConcreteFunction._call_flat(self, args, captured_inputs,␣
 ↪cancellation_manager)
   1753 possible_gradient_type = gradients_util.PossibleTapeGradientTypes(args)
   1754 if (possible_gradient_type == gradients_util.POSSIBLE_GRADIENT_TYPES_NONE
   1755     and executing_eagerly):
   1756   # No tape is watching; skip to running the function.
-> 1757   return self._build_call_outputs(self._inference_function.call(
   1758       ctx, args, cancellation_manager=cancellation_manager))
   1759 forward_backward = self._select_forward_and_backward_functions(
   1760     args,
   1761     possible_gradient_type,
   1762     executing_eagerly)
   1763 forward_function, args_with_tangents = forward_backward.forward()
```

```
File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
 ↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\tensorflow\python\eager\
 ↪py:381, in _EagerDefinedFunction.call(self, ctx, args, cancellation_manager)
    379 with _InterpolateFunctionError(self):
    380   if cancellation_manager is None:
--> 381     outputs = execute.execute(
    382         str(self.signature.name),
    383         num_outputs=self._num_outputs,
    384         inputs=args,
    385         attrs=attrs,
    386         ctx=ctx)
    387   else:
    388     outputs = execute.execute_with_cancellation(
    389         str(self.signature.name),
    390         num_outputs=self._num_outputs,
   (…)
    393         ctx=ctx,
    394         cancellation_manager=cancellation_manager)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.
 ↪10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\tensorflow\python\eager\
 ↪py:52, in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     50 try:
     51   ctx.ensure_initialized()
---> 52   tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name
     53                                       inputs, attrs, num_outputs)
     54 except core._NotOkStatusException as e:
     55   if name is not None:

KeyboardInterrupt:
```