```
import tensorflow as tf
import numpy as np
import gym

# Load cartpole environment
env = gym.make('CartPole-v0')

gamma = 0.99
learning_rate = 0.01
state_size = 4
num_actions = 2
hidden_size = 8
total_episodes = 5000  # Set total number of episodes to train agent on.
max_ep = 999
update_frequency = 5
is_visualize = False

def discount_rewards(r):
  """ take 1D float array of rewards and compute discounted reward """
  discounted_r = np.zeros_like(r)
  running_add = 0
  for t in reversed(range(0, r.size)):
    running_add = running_add * gamma + r[t]
    discounted_r[t] = running_add
  return discounted_r

class PolicyNetworks(tf.keras.Model):
  def __init__(self):
    super(PolicyNetworks, self).__init__()
    self.hidden_layer_1 = tf.keras.layers.Dense(hidden_size, activation='relu')
    self.output_layer = tf.keras.layers.Dense(num_actions, activation='softmax')

  def call(self, x):
    H1_output = self.hidden_layer_1(x)
    outputs = self.output_layer(H1_output)

    return outputs

def pg_loss(outputs, actions, rewards):
  indexes = tf.range(0, tf.shape(outputs)[0]) * tf.shape(outputs)[1] + actions
  responsible_outputs = tf.gather(tf.reshape(outputs, [-1]), indexes)

  loss = -tf.reduce_mean(tf.math.log(responsible_outputs) * rewards)

  return loss
```

Saved successfully! ×

```
ing_rate)

def train_step(model, states, actions, rewards):
  with tf.GradientTape() as tape:
    outputs = model(states)
    loss = pg_loss(outputs, actions, rewards)
  gradients = tape.gradient(loss, model.trainable_variables)
  optimizer.apply_gradients(zip(gradients, model.trainable_variables))

# Declare Policy Gradient Networks
PG_model = PolicyNetworks()

i = 0
total_reward = []
total_length = []

# train start
while i < total_episodes:
  s = env.reset()
  running_reward = 0
  ep_history = []

  for j in range(max_ep):
    if is_visualize == True:
      env.render()
    # Probabilistically pick an action given our network outputs.
    s = np.expand_dims(s, 0)
    a_dist = PG_model(s).numpy()
    a = np.random.choice(a_dist[0], p=a_dist[0])
    a = np.argmax(a_dist == a)
```

```
      s1, r, d, _ = env.step(a)  # Get reward and next state
      ep_history.append([s, a, r, s1])
      s = s1
      running_reward += r

      if d == True:
        ep_history = np.array(ep_history)
        ep_history[:, 2] = discount_rewards(ep_history[:, 2])

        # Make state list to numpy array
        np_states = np.array(ep_history[0, 0])
        for idx in range(1, ep_history[:, 0].size):
          np_states = np.append(np_states, ep_history[idx, 0], axis=0)

        # Update the network parameter
        if i % update_frequency == 0 and i != 0:
          train_step(PG_model, np_states, ep_history[:, 1], ep_history[:, 2])

        total_reward.append(running_reward)
        total_length.append(j)
        break

    # Print last 100 episode's mean score
    if i % 100 == 0:
      print(np.mean(total_reward[-100:]))
    i += 1
```

```
<ipython-input-53-bba22c6d4444>:84: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tupl
  ep_history = np.array(ep_history)
14.0
28.36
32.97
40.33
48.04
37.38
42.99
67.76
62.87
36.61
36.32
36.97
56.17
100.37
113.89
95.48
64.73
96.71
```

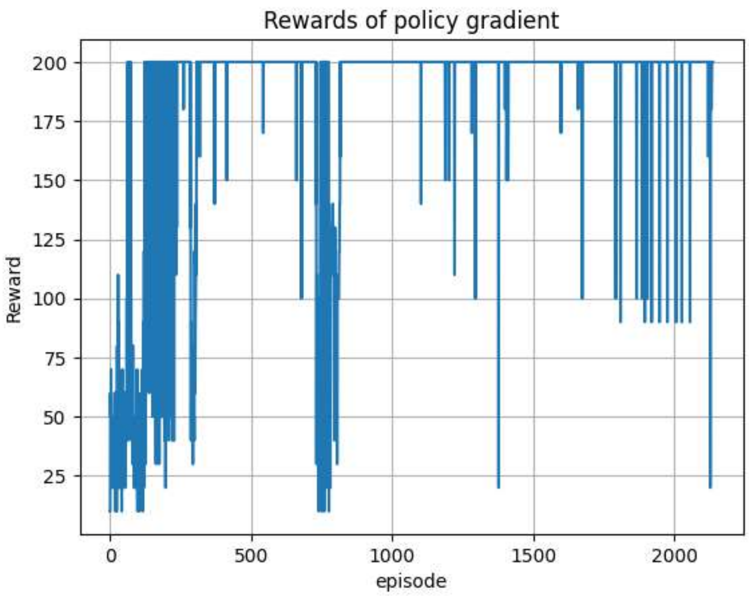
Saved successfully!

```
64.36
158.5
182.78
183.9
192.02
190.56
161.07
56.08
71.56
81.91
94.96
92.61
96.14
159.24
197.09
195.93
196.8
189.17
185.5
175.52
157.13
186.29
195.07
190.3
190.25
187.21
162.4
175.18
177.29
```

```
print(total_reward)
```

```
[14.0, 39.0, 27.0, 15.0, 22.0, 26.0, 14.0, 10.0, 13.0, 25.0, 12.0, 19.0, 22.0, 13.0, 24.0, 24.0, 24.0, 16.0, 15.0, 38.0, 32.0, 11.0, 66.
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell`
  and should_run_async(code)
```

```
import matplotlib.pyplot as plt
total_rewards = [i for i in total_reward if i % 10 == 0]
plt.plot(total_rewards)
plt.xlabel('episode')
plt.ylabel('Reward')
plt.title('Rewards of policy gradient')
plt.grid(True)
plt.show()
```



Saved successfully!

Colab paid products - Cancel contracts here

✓ 0s completed at 11:06 PM