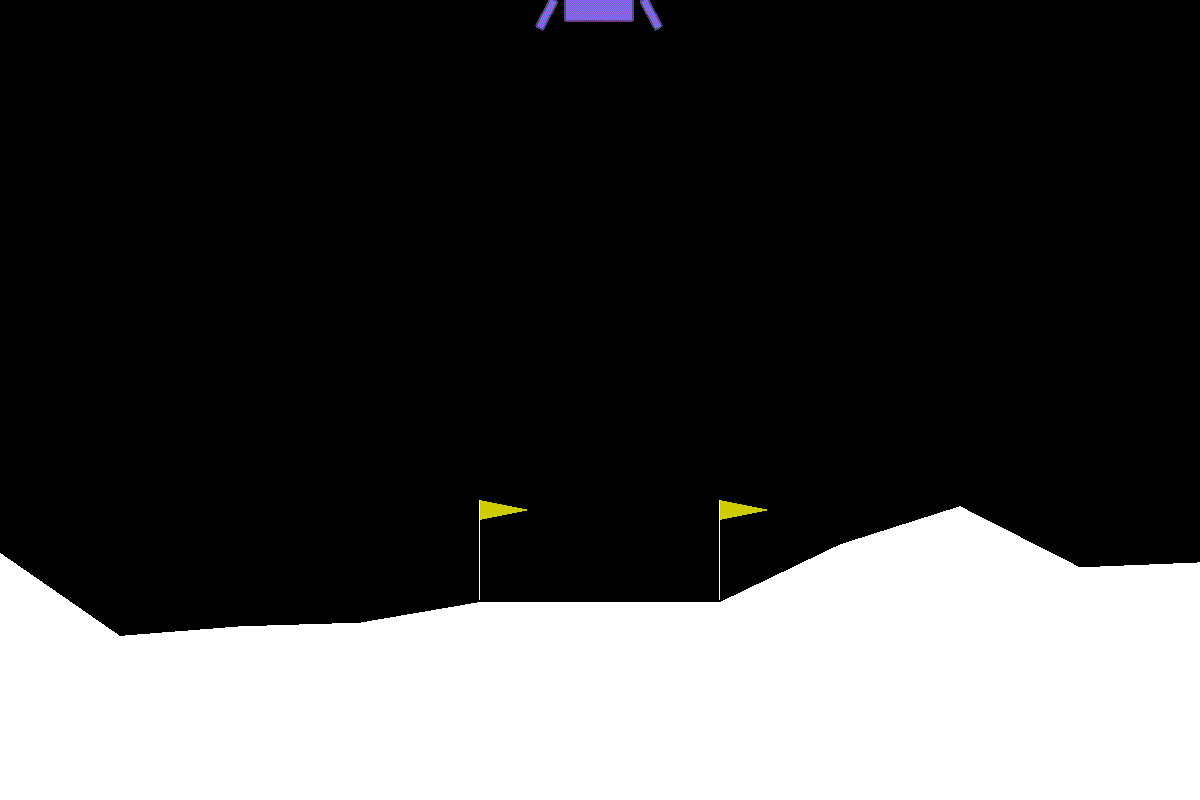
**强化学习课程实验1：LunarLander-DQN**

1. **实验环境**

|  |  |
| --- | --- |
| 系统版本 | Windows10 |
| CUDA版本 | 11.2 |
| CUDNN版本 | 8.1.1 |
| Python版本 | 3.8 |
| requirements | tensorflow-gpu==2.10.0 |
| gym==0.26.2 |
| numpy==1.24.2 |
| pandas==1.5.3 |
| matplotlib==3.7.1 |

1. **控制问题**

Gym-Lunar Lander (discrete version)



This environment is a classic rocket trajectory optimization problem. According to Pontryagin’s maximum principle, it is optimal to fire the engine at full throttle or turn it off. This is the reason why this environment has discrete actions: engine on or off.

——Gym Documentation

1. **算法**

采用DQN，Deep Q-Learning Network作为控制算法。

1. DQN创新点：

基于Q-Learning构造Loss Function。

通过experience replay（经验池）解决相关性及非静态分布问题；

使用TargetNet解决稳定性问题。

1. DQN优点：

算法通用性，可玩不同游戏；

End-to-End 训练方式；

可生产大量样本供监督学习。

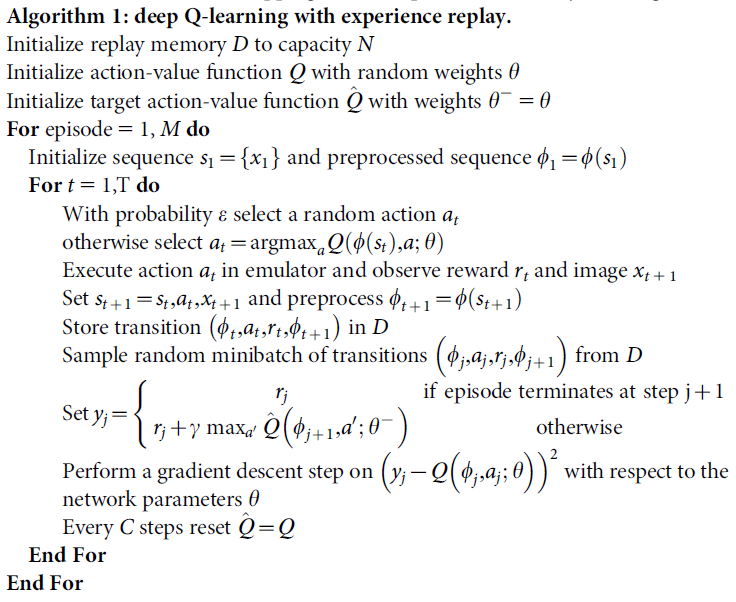
1. DQN缺点：

无法应用于连续动作控制；

只能处理只需短时记忆问题，无法处理需长时记忆问题；

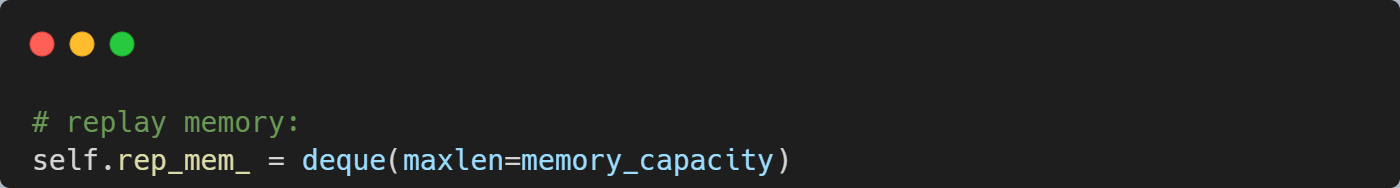
CNN不一定收敛，需精良调参。

1. **算法详细说明与实现**
2. 算法伪代码

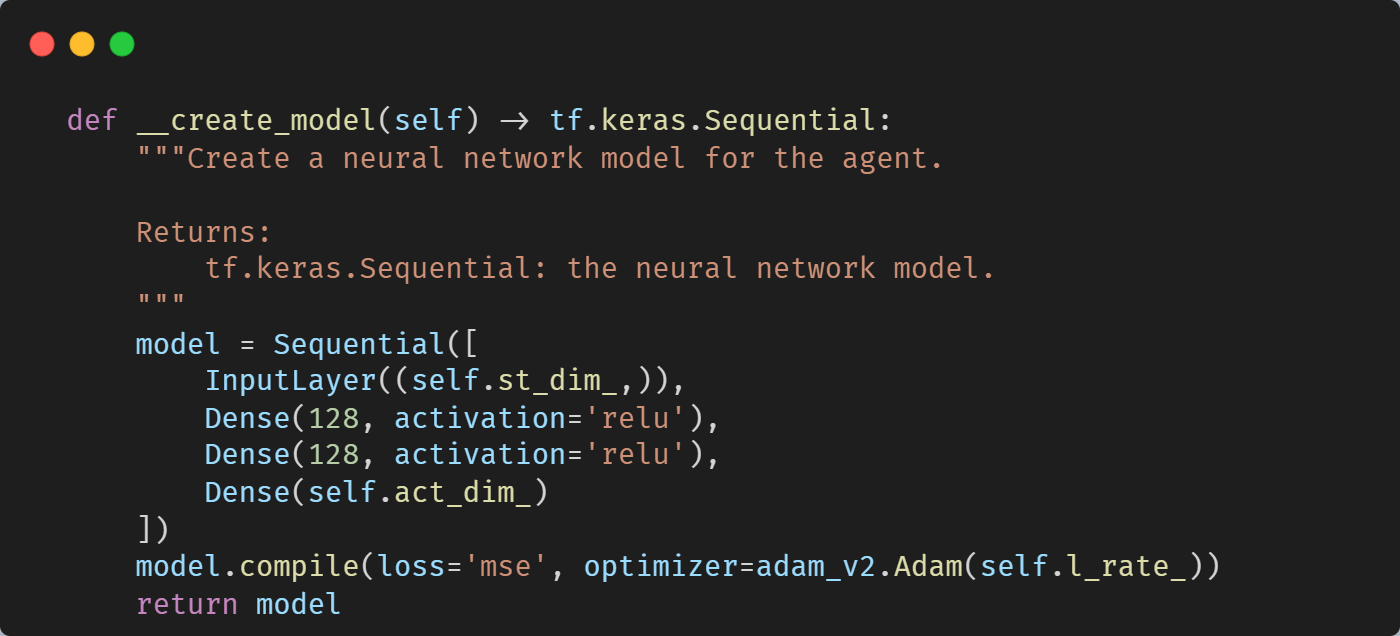


1. 算法实现

DQN算法作为Agent类中的一个方法。根据以上伪代码，逐条在Python中实现。

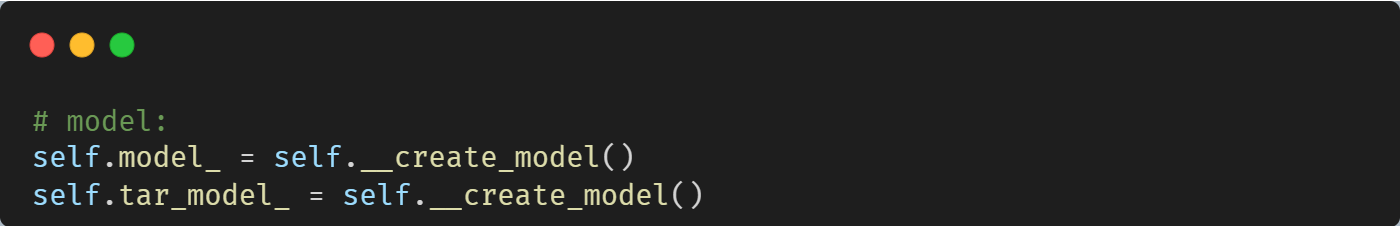
使用一个队列deque来作为经验池：

使用tensorflow.keras构建神经网络，封装为create\_model()函数：



有两个隐含层，神经元数目根据情况调整，使用MSE损失函数、Adam优化器。

分别创建两个网络：Q-network和target Q-network：



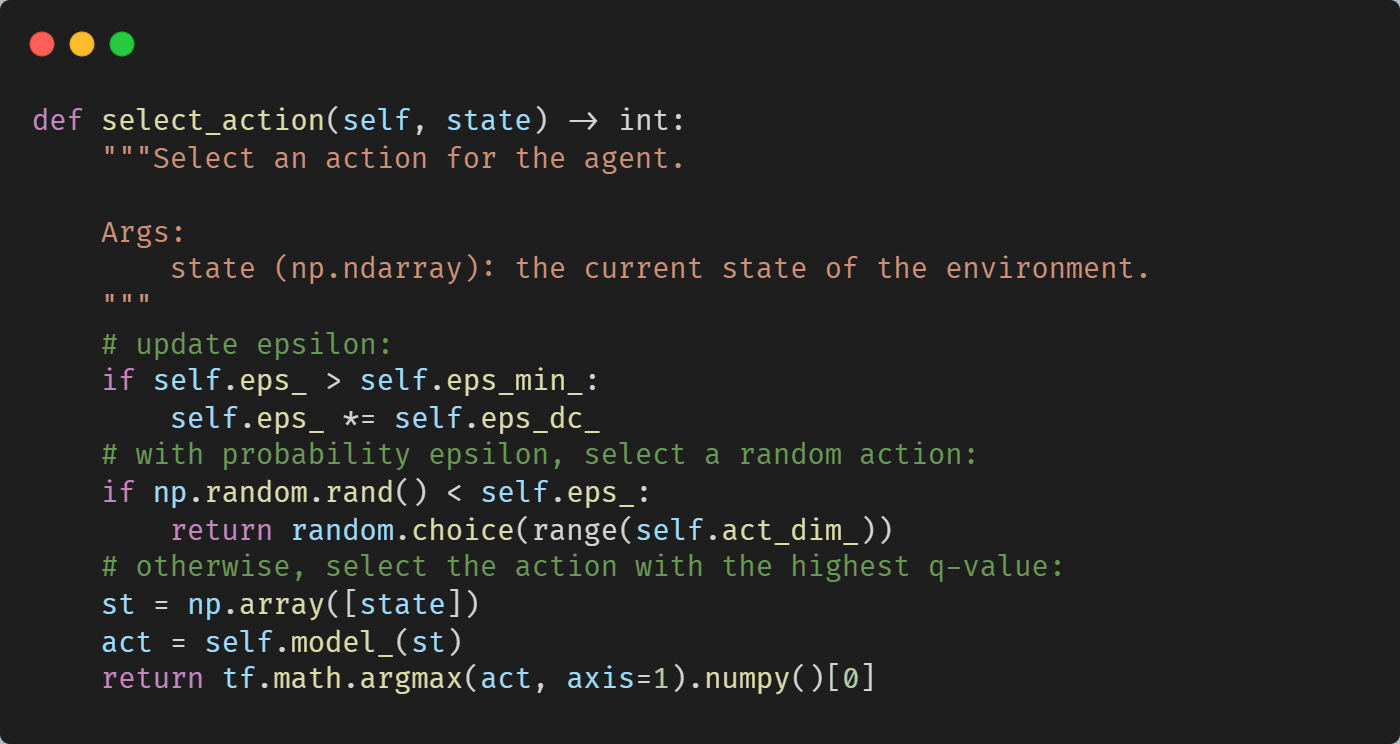
第一层循环，设置训练进行的回合数。

环境重置，并获取初始状态：



第二层循环，可使用while True也可以设置单回合最大步数。

获取0-1之间随机数，小于ε采取随机动作，否则根据网络采取能得到最大Q值的动作，并且应当随训练进行不断更新ε的值，使随机动作的概率越来越小（使用乘一个衰减值的方法）。封装为select\_action()函数：



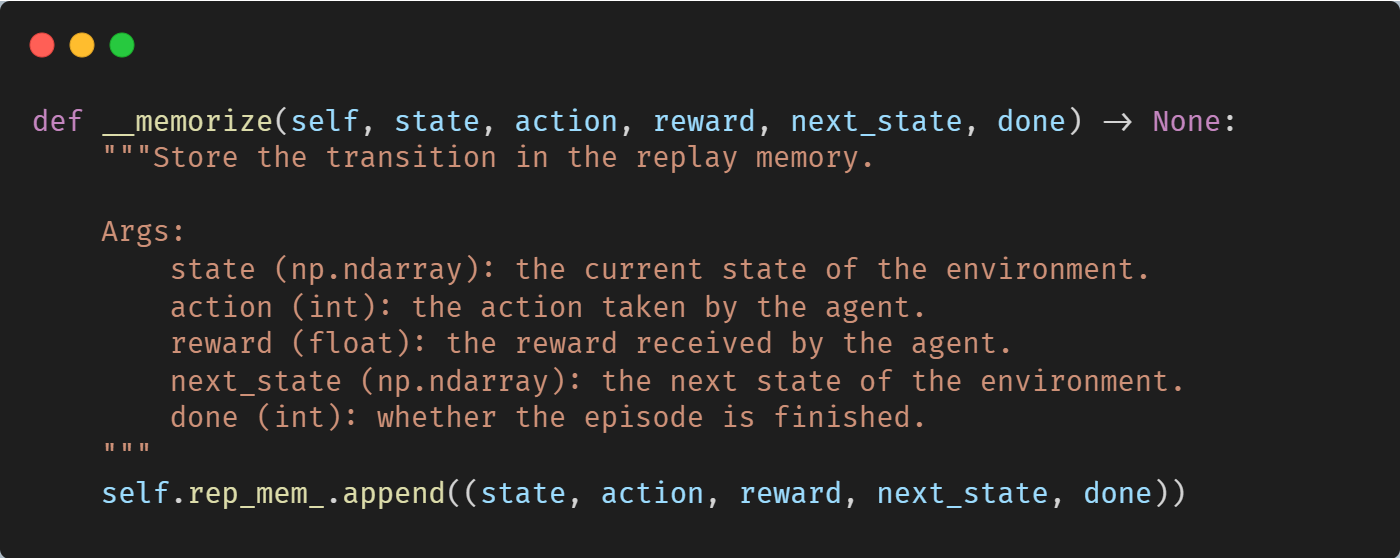
在循环中调用：



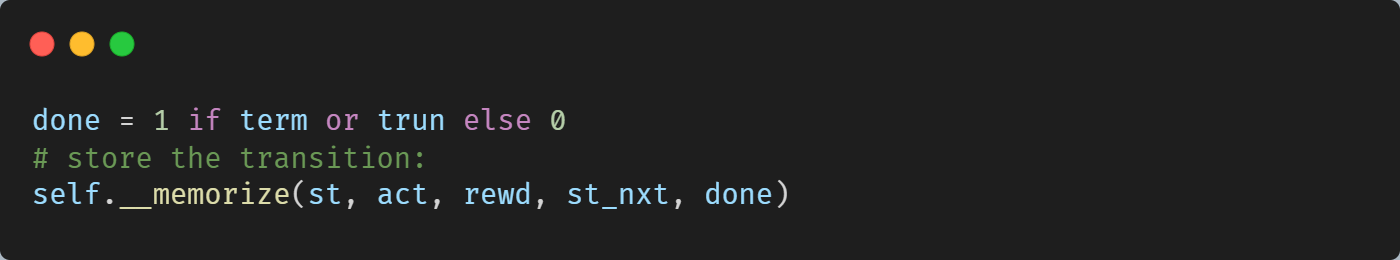
将上一步得到的动作作用到环境中，获得新的状态、奖励等：



记忆过程，将当前状态、采取动作、奖励、下一状态以及是否结束的值存储到经验池deque中，封装为memorize()函数：



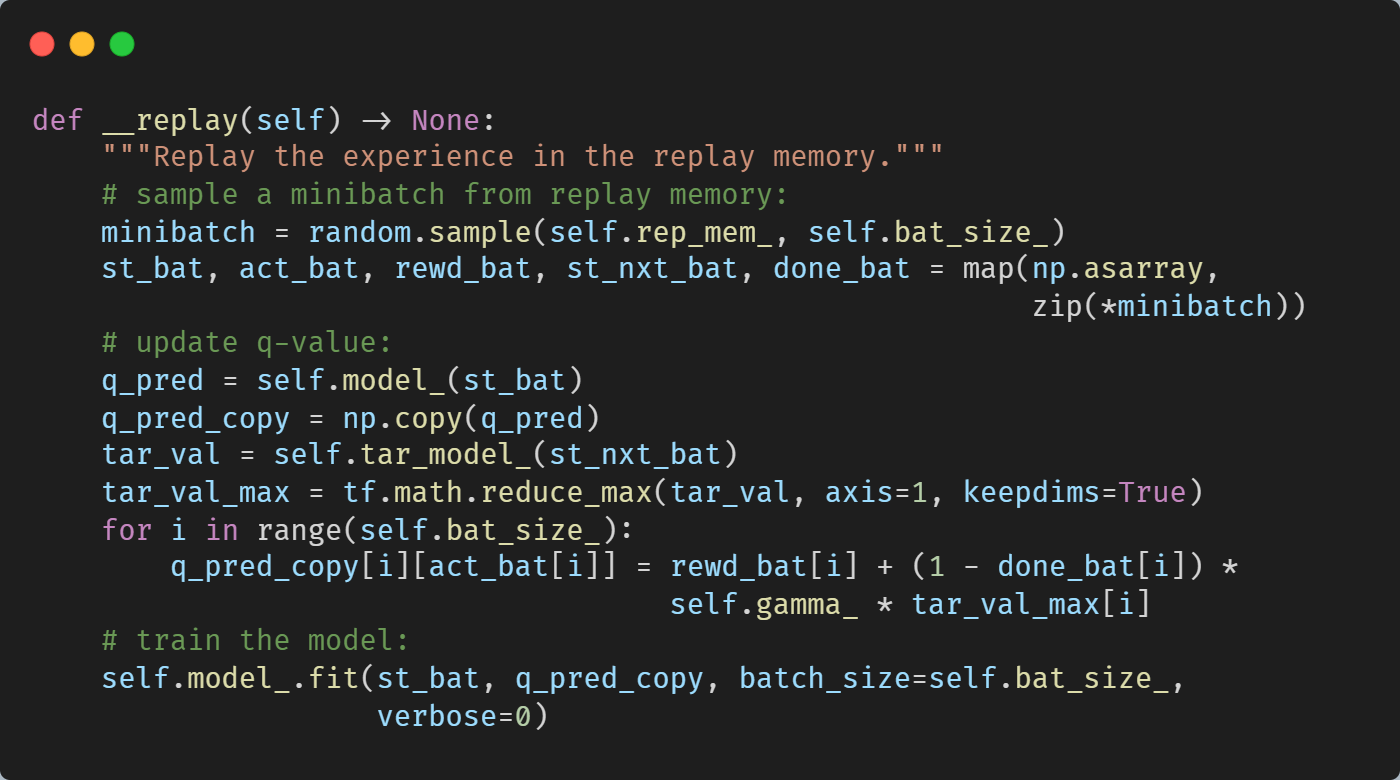
在循环中调用：



之后记忆完成后可以进行状态的更新：

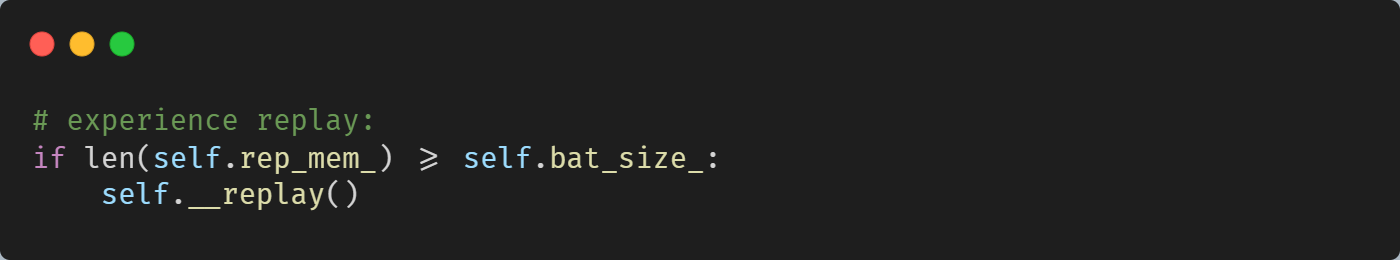


以上几步均属于经验重放过程，从经验池中随机采样一定组的数据，训练更新网络，封装为replay()函数：

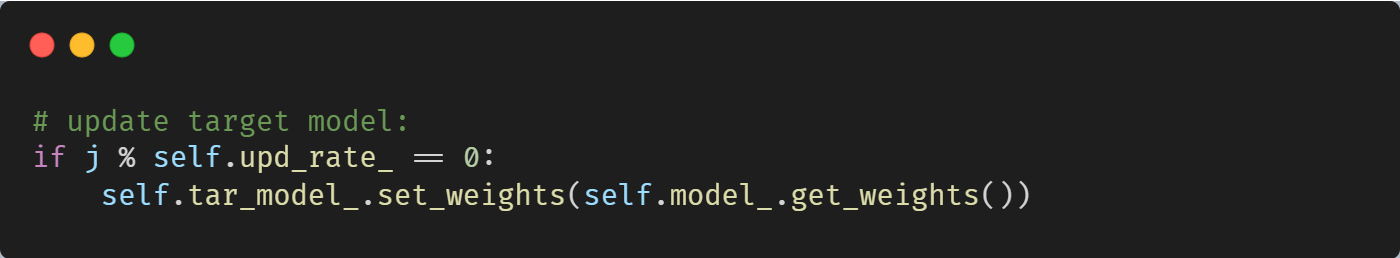


这里关于折扣因子γ还有一点疑惑，具体写在第六部分。

在循环中调用：

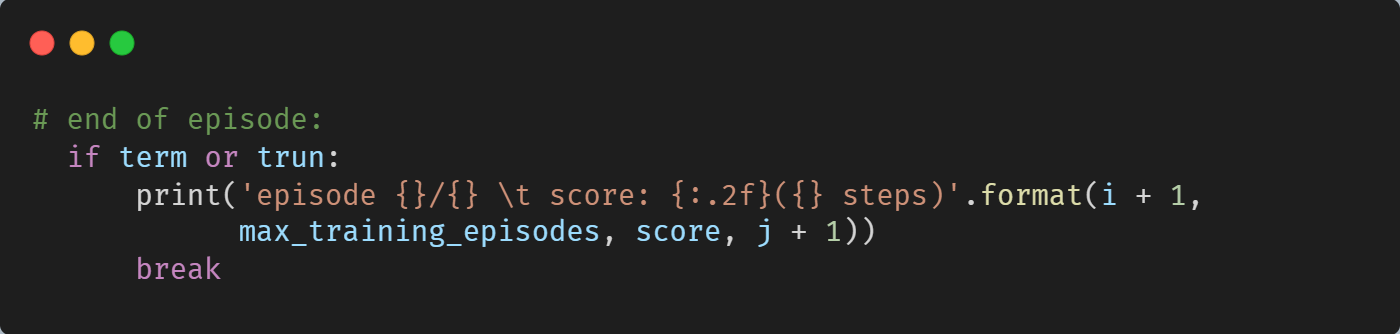


一定步数之后对target Q-network进行硬更新，即将Q-network的权重赋值过去：



1. 设定终止条件

当一个回合触发结束条件后，包括飞船接触地面、已执行的步数超过内置的最大步数（1000），进入到下一回合的训练，打印这一回合的结果：



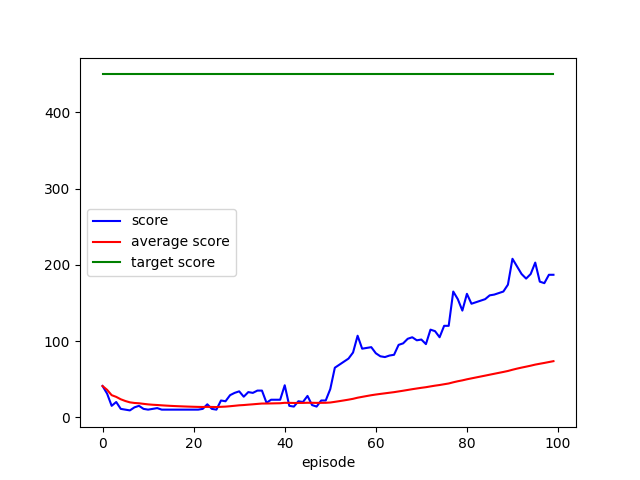
1. 程序整体代码

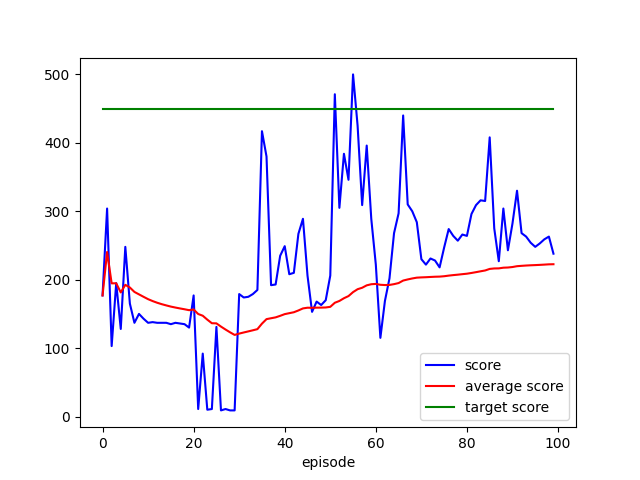
详见“./LunarLander- DQN/”。

1. **训练结果**

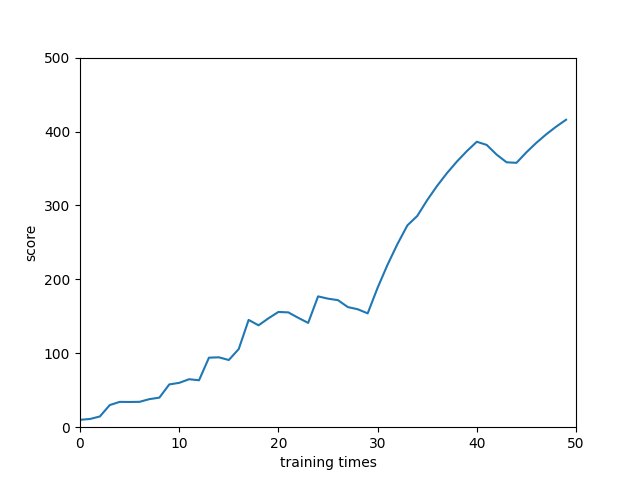
多次训练的结果并不是很理想，均分始终在-100左右，推测是训练次数仍然不够，状态空间过大，从得分上看，甚至没有平稳降落的情况。于是改用较为简单CartPole来测试、训练，结果如下：

先后各进行了100次的训练，蓝色曲线是一个回合的得分，红色是当前平均得分：

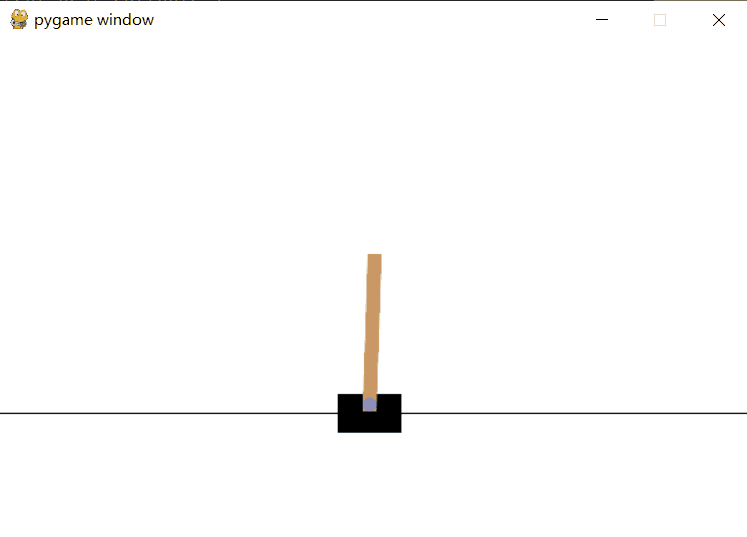




下面是使用策略梯度在CartPole环境下的训练结果：



当前已经能够在每次测试中稳定保持到最大步数限制（500）：



而且策略梯度训练速度更快，占用内存更少，不太清楚应当如何改进DQN来得到更好的效果。

1. **遇到的困难、误区&解决方法**
2. TensorFlow无法识别GPU的问题
   1. TensorFlow文档中标明当前最新版本2.11.0不支持Windows平台下调用GPU，需要降级到2.10.0，pip install tensorflow-gpu==2.10.0
   2. TensorFlow、CUDA、CUDNN的版本应当对应，参考官方文档：https://tensorflow.google.cn/install/source?hl=en#gpu
3. 在前面提到的折扣因子的问题

在做像CartPole这种不断给出大于零的奖励的情形下，我可以理解折扣因子在0-1之间，是为了强化当前得到的奖励。但是对于LunarLander这样的reward基本全为负（消耗燃料扣分、坠毁扣分，只有成功落地才加分）的情况，折扣因子还应该是在0-1范围吗，这样是不是相当于减少了未来的扣分，起到相反的效果？

1. 对单回合的步数应当做出一定限制

在多次实验中可以看到，过少的步数（比如100以下）的结果必然是由于操作偏差较大飞船快速坠毁；而过多的步数（比如500以上）的结果是智能体避免坠毁造成的扣分而在空中悬停，而不是降落，这显然不是想要得到的结果。并且查看了很多其他人所得到的较好的训练结果，步数一般在200-300之间能够平稳降落得到更高的分数，所以人为对奖励值做出一些调整，设定两个阈值，小于步数下限时结束和大于步数上限时未结束都将reward设置为惩罚值。

1. epsilon值的调整

因为内存有限的原因，每次训练的回合数都设置的较低，比如50回合，保存、加载模型，分多次进行训练，但训练过程中epsilon的值应当是不断衰减的，而这样多次训练会因为没有去手动调整epsilon的值，而采取过多的随机策略。所以应当再每次训练时手动调整epsilon到衰减后的值。

1. 效果仍然不理想