强化学习导论

实验报告



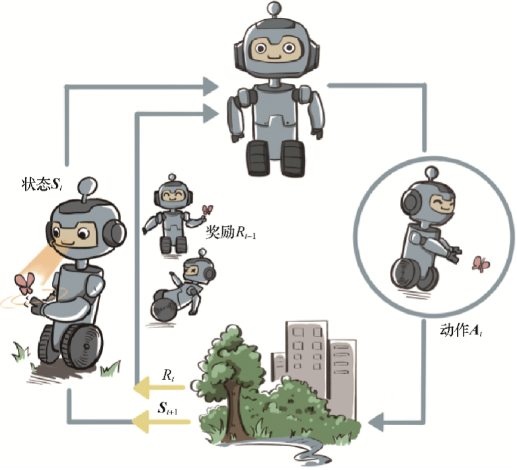
|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | **柳成林** | **学号** | **20206722** |
| **班级** | **软件2001** | **任课教师** | **毛克明** |
| **课程名称** | **强化学习导论** | | |
| **开设学期** | **2022-2023 春季学期** | | |
| **开设时间** |  | | |
| **报告日期** | **2023/5/04** | | |

**东北大学软件学院**

# 1.主要内容

## 1.1 内容介绍

强化学习（reinforcement learning）是一种机器学习方法，它通过试错来训练智能体（agent）在特定情境下作出最优的决策。



图表 1 强化学习示例

我们提出了第一个利用强化学习直接从高维感官输入成功学习控制策略的深度学习模型（Deep Q-Network，）。该模型是一个卷积神经网络，用Q-learning的一个变种进行训练，其输入是原始像素，其输出是一个估计未来奖励的价值函数。我们将我们的方法应用于街机学习环境中的七个Atari 2600游戏，没有调整结构或学习算法。我们发现，它在其中6个游戏上的表现超过了以前所有的方法，在其中3个游戏上超过了人类专家。

## 1.2 意义

学习直接从视觉和语音等高维感官输入中控制代理是强化学习（RL）的长期挑战之一。在这些领域操作的大多数成功的RL应用都依赖于手工制作的特征与线性值函数或策略表示相结合。显然，这种系统的性能在很大程度上依赖于特征表示的质量。

深度学习的最新进展使得从原始感官数据中提取高级特征成为可能，从而在计算机视觉和语音识别方面取得了突破。这些方法利用了一系列的神经网络架构，包括卷积网络、多层感知器、受限玻尔兹曼机和递归神经网络，并利用了监督和无监督学习。似乎很自然地要问，类似的技术是否也能对感官数据的RL有益。

## 1.3 难点

深度学习的角度来看，强化学习提出了几个挑战。首先，到目前为止，大多数成功的深度学习应用都需要大量的标记训练数据。另一方面，RL算法必须能够从标量奖励信号中学习，该信号经常是稀疏的、有噪声的和延迟的。与监督学习中发现的输入和目标之间的直接关联相比，行动和结果奖励之间的延迟可能长达数千个时间段(timesteps)，似乎特别令人生畏。另一个问题是，大多数深度学习算法假设数据样本是独立的，而在强化学习中，人们通常会遇到高度相关的状态序列。此外，在RL中，数据分布随着算法学习新的行为而改变，这对于假设有固定基础分布的深度学习方法来说是有问题的。

本文表明，卷积神经网络可以克服这些挑战，在复杂的RL环境中从原始视频数据中学习成功的控制策略。该网络是用Q-learning算法的一个变种来训练的，用随机梯度下降来更新权重。为了缓解相关数据和非平稳分布的问题，我们使用了一种experience replay mechanism，该机制对以前的转换(transitions)进行随机采样，从而使训练分布在许多过去的行为中变得平滑。

## 1.4 实现

我们将我们的方法应用于一系列在Arcade Learning Environment（ALE）中实现的Atari 2600游戏。Atari 2600是一个具有挑战性的RL测试平台，它为agents提供了高维度的视觉输入（210×160 RGB视频，60Hz）和一系列多样化的、有趣的任务，这些任务被设计成对人类玩家来说是困难的。我们的目标是创建一个单一的神经网络代理，能够成功地学习玩尽可能多的游戏。

网络没有被提供任何游戏特定的信息或手工设计的视觉特征，也不了解模拟器的内部状态；它只从视频输入、奖励和终端信号以及可能的行动集合中学习，就像人类玩家一样。此外，网络结构和用于训练的所有超参数在所有游戏中都保持不变。到目前为止，该网络在我们尝试的七种游戏中的六种上的表现超过了以前所有的RL算法，在其中三种上超过了人类专家玩家。图1提供了用于训练的五个游戏的截图样本。



图表 2 Atari 游戏示例

# 2.相关方法

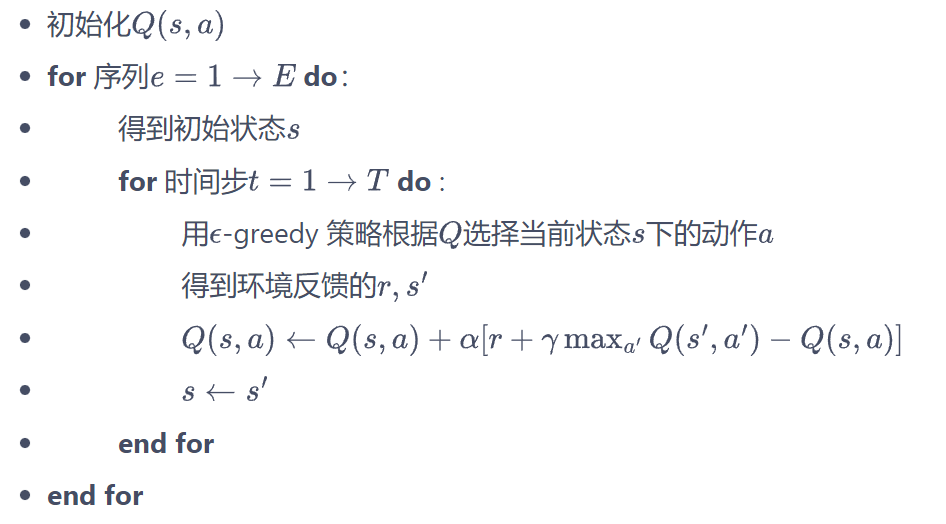
## 2.1 Q-Learning

Q-Learning是一种基于时序差分的强化学习算法，时序差分算法的优势是在不了解学习环境的条件下也适用。在已知环境的情况下，我们一般采用动态规划算法。动态规划算法要求马尔可夫决策过程是已知的，即要求与智能体交互的环境是完全已知的（例如迷宫或者给定规则的网格世界）。在此条件下，智能体其实并不需要和环境真正交互来采样数据，直接用动态规划算法就可以解出最优价值或策略。

但这在大部分场景下并不现实，机器学习的主要方法都是在数据分布未知的情况下针对具体的数据点来对模型做出更新的。对于大部分强化学习现实场景（例如电子游戏或者一些复杂物理环境），其马尔可夫决策过程的状态转移概率是无法写出来的，也就无法直接进行动态规划。在这种情况下，智能体只能和环境进行交互，通过采样到的数据来学习。

时序差分是一种用来估计一个策略的价值函数的方法，它结合了蒙特卡洛和动态规划算法的思想。时序差分方法和蒙特卡洛的相似之处在于可以从样本数据中学习，不需要事先知道环境；和动态规划的相似之处在于根据贝尔曼方程的思想，利用后续状态的价值估计来更新当前状态的价值估计。

其算法流程为：



图表 3 Q-Learning算法的流程

Q-Learning的弊端非常明显，由于要维护一个离散的表格，当处于复杂的环境时，表格需要的空间很大，这就会造成巨大的空间和时间损失。而本文提出的深度强化学习算法（DQN）对传统 Q-Learning 的改进之处在于：使用神经网络实现 Q值函数逼近，避免了动作空间非常大时需要存储巨大的 Q表格等问题；采用经验重放技术，利用之前的经验随机训练神经网络，增加样本的使用效率和泛化性能。

在提出DQN之前，早期较为成功的强化学习案例使用Q-learning较多。

## 2.2 相关工作

也许强化学习最有名的成功案例是TD-gammon，一个完全通过强化学习和自我发挥来学习的双陆棋程序，并达到了超越人类的水平。TD-gammon使用了一种类似于Q-learning的无模型强化学习算法，并使用一个隐藏层的多层感知器来逼近价值函数。

然而，早期对TD-gammon的后续尝试，包括对国际象棋、围棋和跳棋的相同方法的应用都不太成功。这导致人们普遍认为，TD-gammon方法是一个特例，只在双陆棋中有效，也许是因为掷骰子的随机性有助于探索状态空间，也使价值函数特别平滑。

此外，有研究表明，将无模型的强化学习算法，如Q-learning与非线性函数近似器，或实际上与off-policy学习相结合，可能会导致Q网络的分歧（diverge）。随后，强化学习的大部分工作集中在具有更好收敛保证的线性函数近似器上。

最近，人们对深度学习与强化学习的结合又有了新的兴趣。深度神经网络被用来估计环境E；受限玻尔兹曼机被用来估计价值函数；或策略。此外，Q-learning的发散问题已经被梯度时间差法（gradient temporal-difference methods）部分解决。这些方法被证明在用非线性函数近似器评估固定策略时收敛；或者在用Q-learning的限制性变体学习线性函数近似的控制策略时收敛。然而，这些方法还没有被扩展到非线性控制。

也许与我们的方法最相似的前期工作是神经拟合Q-learning（NFQ）。NFQ优化方程2中的损失函数序列，使用RPROP算法来更新Q-网络的参数。然而，它使用的是批量更新，每次迭代的计算成本与数据集的大小成正比，而我们考虑的是随机梯度更新，每次迭代的恒定成本很低，并可扩展到大数据集。NFQ也被成功地应用于使用纯视觉输入的简单现实世界的控制任务，首先使用深度自动编码器来学习任务的低维表示，然后将NFQ应用于该表示。

相比之下，我们的方法是直接从视觉输入中进行端到端的强化学习；因此，它可以学习与分辨动作值直接相关的特征。Q-learning以前也曾与experience replay和一个简单的神经网络相结合，但同样是从低维状态而不是原始视觉输入开始。

使用Atari 2600模拟器作为强化学习平台是由介绍的，他应用标准的强化学习算法与线性函数近似和通用视觉特征。随后，通过使用更多的特征，并使用tug-of-war hashing将特征随机投射到一个较低维度的空间，结果得到了改善。HyperNEAT进化架构也被应用于Atari平台，它被用来进化（分别为每个不同的游戏）一个代表该游戏策略的神经网络。当使用模拟器的重置设施针对确定性序列反复训练时，这些策略能够利用几个雅达利游戏的设计缺陷。

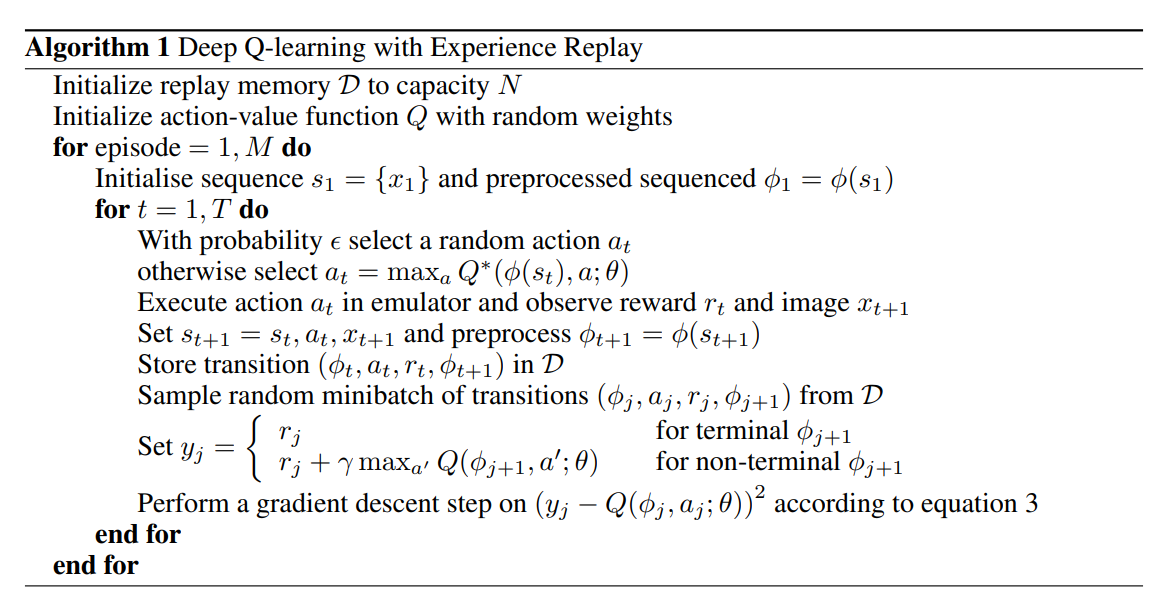
# 3.主要方法

## 3.1c主体流程

最近在计算机视觉和语音识别方面的突破依赖于在非常大的训练集上有效训练深度神经网络。最成功的方法是直接从原始输入进行训练，使用基于随机梯度下降的轻型更新。通过向深度神经网络输入足够的数据，通常可以学到比手工制作的特征更好的表征。这些成功的经验促使我们采取强化学习的方法。我们的目标是将强化学习算法连接到一个深度神经网络，该网络直接在RGB图像上运行，并通过使用随机梯度更新来有效地处理训练数据。

Tesauro的TD-Gammon架构为这种方法提供了一个出发点。这个架构直接从算法与环境的互动（或者在双陆棋的情况下，通过self-play游戏）中提取的经验的 on-policy samples样本st,at,rt,st+1,at+1来更新估计价值函数的网络参数。由于这种方法在20年前就能超越最好的人类双陆棋手，我们自然会想，20年的硬件改进，加上现代深度神经网络架构和可扩展的RL算法，是否会产生重大进展。

与TD-Gammon和类似的在线方法相比，我们利用一种被称为**experience replay**的技术，我们将agent在每个时间步长的经验et=( st,at,rt,st+1)存储在一个数据集D=e1,,,eN中，通过许多episode汇集到一个replay memory。在算法的内循环中，我们对经验样本e ∼ D应用Q-learning更新，或称minibatch更新，这些样本是从存储样本池中随机抽取的。在进行经验回放后，agent根据贪婪政策选择并执行一个行动。由于使用任意长度的历史作为神经网络的输入是很困难的，所以我们的q函数将工作于由函数ϕ 产生的历史记录的固定长度表示。完整的算法，我们称之为深度q学习，在算法1中提出。



图表 4 DQN算法流程

这种算法是model-free：它直接使用仿真器E的样本来解决强化学习任务，而不明确构建E的估计值。它也是 off-policy：它学习贪婪策略。

这种方法比标准的在线Q-learning有几个优点。首先，每一步经验都有可能用于许多权重更新，这使得数据效率更高。第二，直接从连续的样本中学习是低效的，因为样本之间有很强的关联性；随机化的样本打破了这些关联性，因此减少了更新的方差。第三，在策略学习时，当前参数决定了参数训练的下一个数据样本。例如，如果动作最大化是向左移动，那么训练样本将被来自左侧的样本所控制;如果最大行动向右移动那么训练分布也会向右移动。很容易看到不必要的反馈循环可能会出现，参数可能会卡在一个糟糕的局部最小值，甚至灾难性地发散。

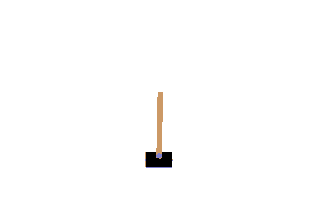
在实践中，我们的算法只在重放存储器中存储最后的N个经验元组，并在执行更新时从D中均匀地随机采样。这种方法在某些方面是有局限性的，因为内存缓冲区不能区分重要的transitions，而且由于有限的内存大小N，总是用最近的transitions来覆盖。同样，统一采样对重放内存中的所有transitions给予同等的重视。一个更复杂的抽样策略可能会强调我们可以从中学习到最多的transitions，类似于prioritized sweeping。

# 4.实验分析

## 4.1 实验背景

论文中使用图像预处理获取atari中的RGB图像，我的代码中将这一部删除，因为我选取的游戏较为简单，如果需要处理图像数据，则需要对Qnet的参数进行修改。将卷积层加入其网络结构以提取图像特征，最终实现以图像为输入的强化学习。DQN 网络通常会将最近的几帧图像一起作为输入，从而感知环境的动态性

以下图中所示的所示的车杆（[CartPole](https://github.com/openai/gym/wiki/CartPole-v0)）环境为例，它的状态值是连续的，动作值是离散的。



图表 5 CartPole环境示意图

在车杆环境中，有一辆小车，智能体的任务是通过左右移动保持车上的杆竖直，若杆的倾斜度数过大，或者车子离初始位置左右的偏离程度过大，或者坚持时间到达 200 帧，则游戏结束。智能体的状态是一个维数为 4 的向量，每一维都是连续的，其动作是离散的，动作空间大小为 2。在游戏中每坚持一帧，智能体能获得分数为 1 的奖励，坚持时间越长，则最后的分数越高，坚持 200 帧即可获得最高的分数。

以下是CartPole的状态环境：

图表 6 状态表格

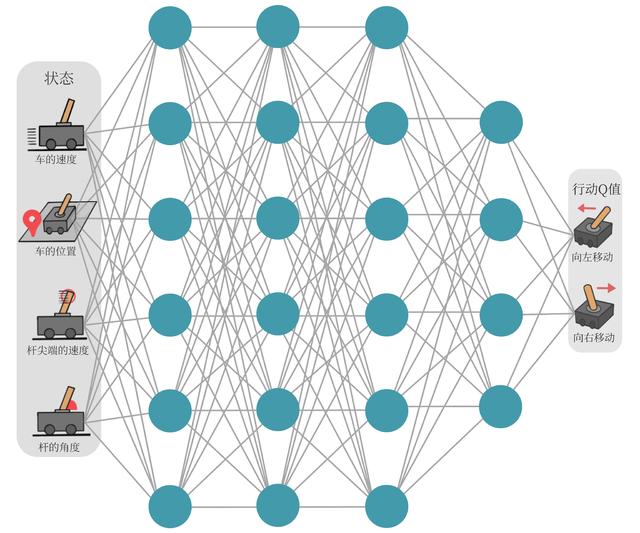
| **维度** | **意义** | **最小值** | **最大值** |
| --- | --- | --- | --- |
| 0 | 车的位置 | -2.4 | 2.4 |
| 1 | 车的速度 | -Inf | Inf |
| 2 | 杆的角度 | ~ -41.8° | ~ 41.8° |
| 3 | 杆尖端的速度 | -Inf | Inf |

以下是CartPole的动作：

图表 7 动作表格

| **标号** | **动作** |
| --- | --- |
| 0 | 向左移动小车 |
| 1 | 向右移动小车 |

工作在CartPole环境中的Q网络示意图：



图表 8 网络示意图

## 4.2 实验内容展示

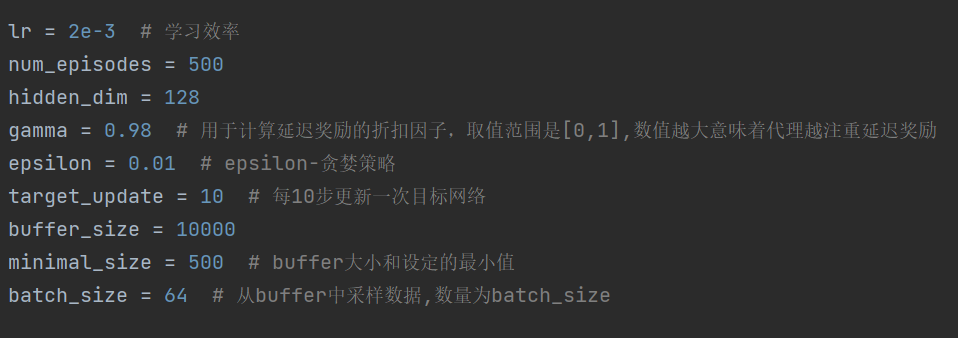
实验环境：

gym

pytorch

python3.9

相关变量定义：



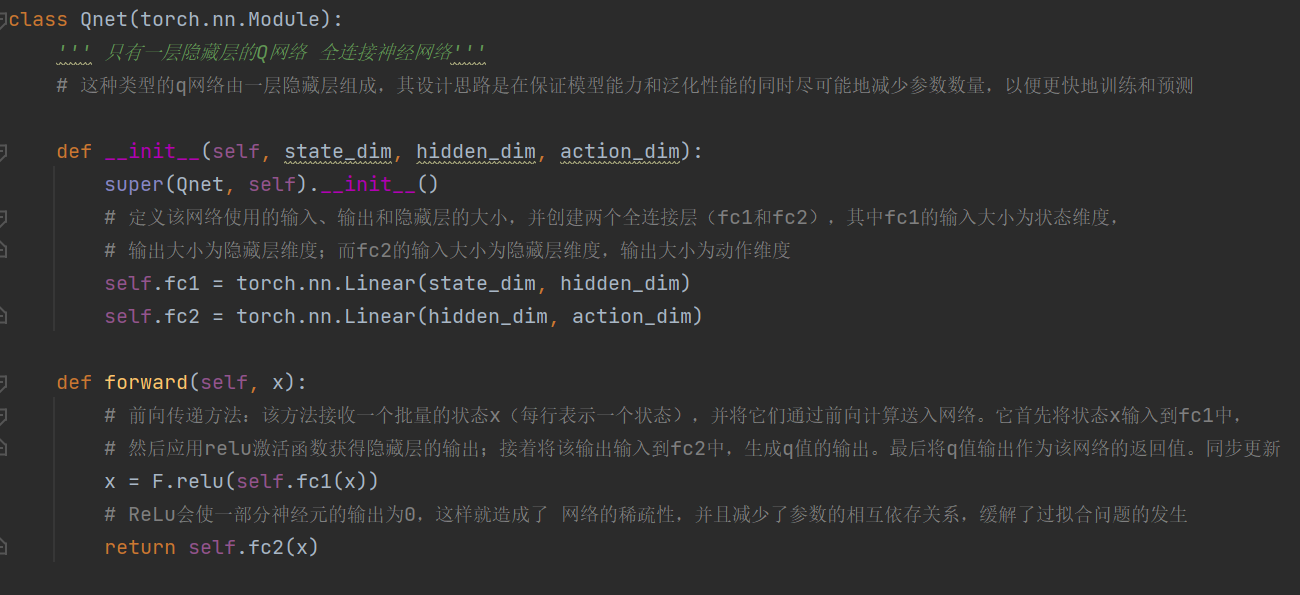
图表 9 实验变量

关键代码编写：

这里解释一下，部分代码并没有完全按照论文的设定，比如目标网络。

1.Qnet网络：只有一层隐藏层的Q网络，用于替代Q表格。

使用一个全连接神经网络，只输入状态值, 输出所有的动作值, 然后按照 Q-learning 的原则, 直接选择拥有最大Q值的动作当做下一步要做的动作。

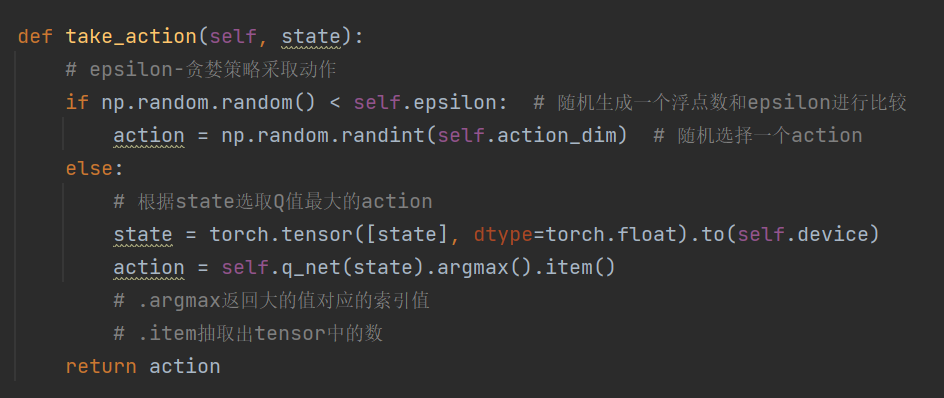


图表 10 Qnet网络

2.epsilon-贪婪策略：该策略主要包含两部分：选择行动的方式和选择行动的依据。

对于选择行动的方式，epsilon-贪婪策略会以1-epsilon的概率选择最优的行动，而以epsilon的概率随机选择其他行动。其中，epsilon通常被设置为一个很小的数，比如0.1。

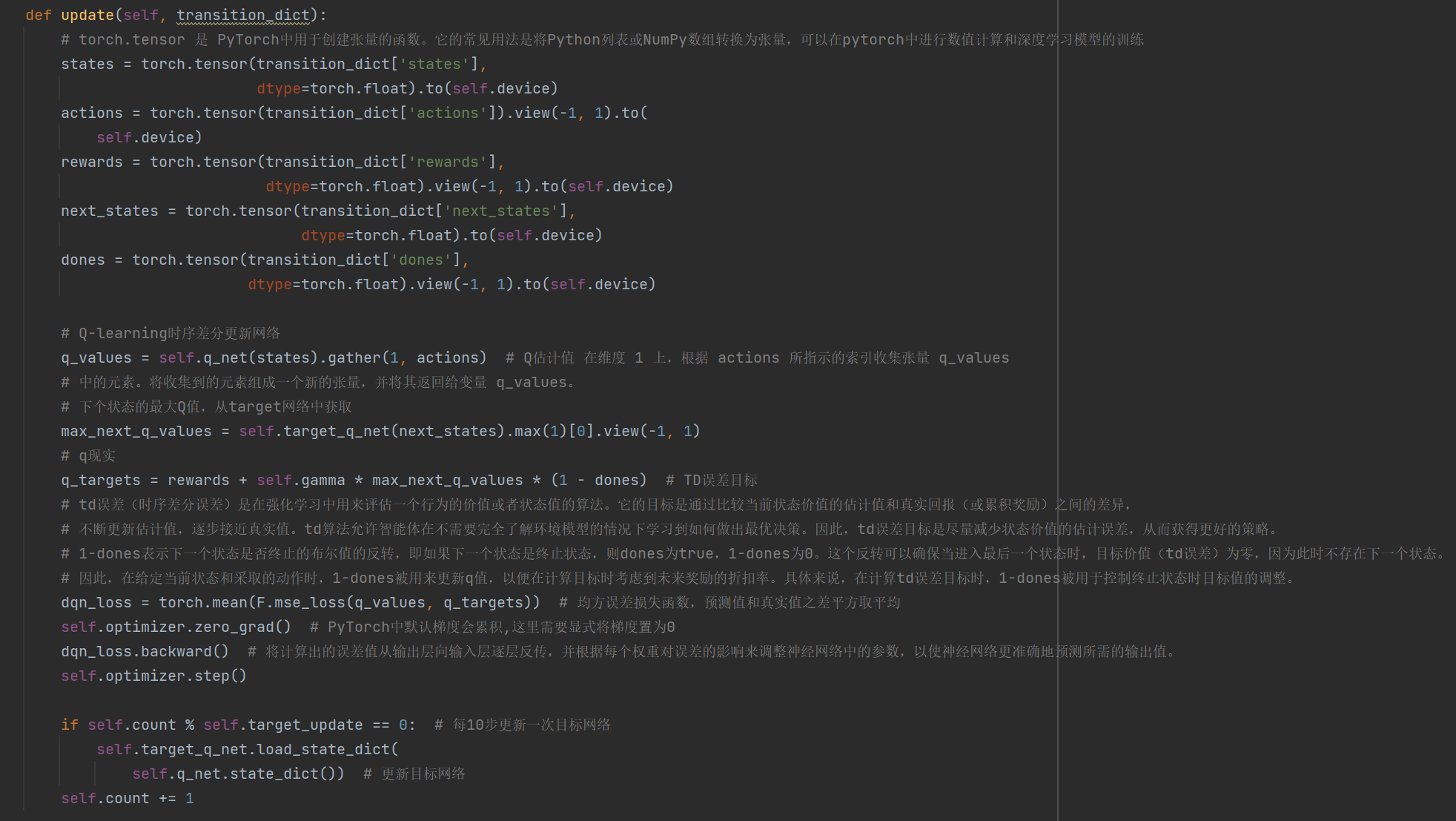
对于选择行动的依据，epsilon-贪婪策略的依据就是当前已知的q值函数。在具体实现时，需要先初始化一个空的q值函数，然后通过与环境的交互来更新q值函数，最后根据epsilon-贪婪策略进行行动选择。



图表 11 take action函数

3.DQN-update方法：这个方法是DQN的一个重要组成部分，用于更新神经网络的参数。具体来说，它接收一个包含一组转换的字典（这些转换代表了智能体从环境中观察到的状态、采取的行动、获得的奖励、下一个状态和终止状态），并将它们转换为张量，然后使用这些张量计算q值和目标q值。接着，它使用均方误差损失函数计算DQN损失，并通过反向传播更新参数。最后，如果计数器达到目标更新值，则使用当前的q网络状态字典更新目标q网络状态字典。

具体地，这个方法开始时将输入数据转换为pytorch张量，并将其移动到当前设备上（gpu或cpu）。然后，它调用q网络计算当前状态下每个可选操作的对应q值，并根据给定的操作选择当前q值。接着，它使用目标q网络计算下一个状态下所有可选操作的最大q值，并为下一个状态的q值目标计算td误差折现值。通过计算当前q值与该目标之间的均方误差，我们可以计算DQN损失。为了训练模型，我们需要使损失最小化。我们使用优化器（例如sgd）在反向传播过程中更新参数，以最小化该损失。最后，当计数器达到预设的目标更新阈值时，我们把当前q网络状态字典复制给目标q网络状态字典，以确保每隔几步更新一次目标网络。我们使用计数器来跟踪当前模型迭代的步数，以便定期更新目标网络。



图表 12 update函数

4.经验回放池：在使用经验回放的过程中，我们会先将观察到的每个状态和对应的行动存储在记忆库中。在之后进行网络训练时，我们可以通过随机采集之前存储的这些状态和行动数据构建小批量输入数据，从而加速网络的训练过程。在模型进行优化时，我们会使用这些样本数据来计算损失并更新权重参数，以提高模型的准确性和普适性。

总的来说，经验回放可以帮助我们更好地利用曾经观察过的数据来提高模型的效率和性能，同时也可以减少需要新数据的需求，节省时间和成本。



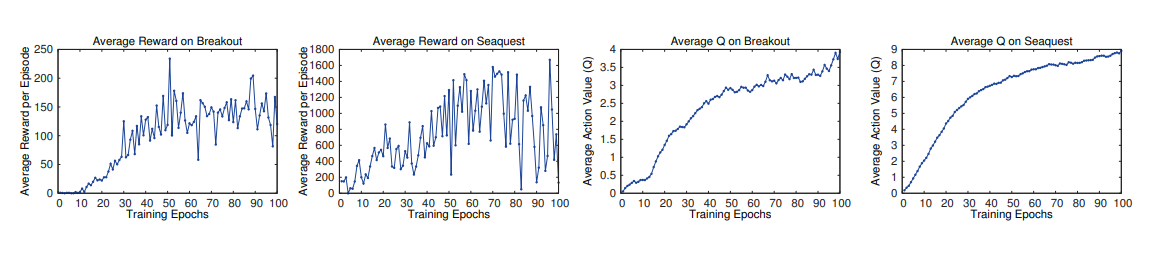
图表 13 经验回放池

## 4.3 实验结果分析

在文章中，作者使用算法对Seaquest和Breakout游戏进行训练，结果如下图所示。

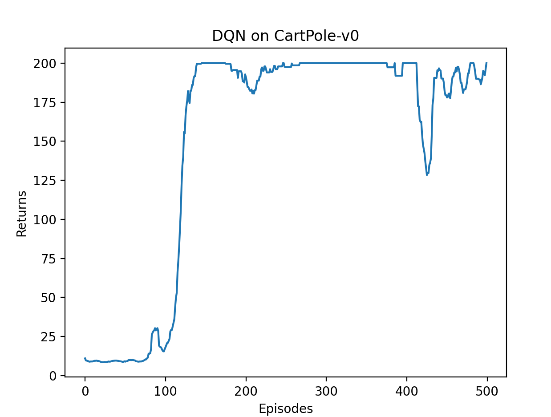
图中最左边的两幅图显示了在Seaquest和Breakout游戏的训练过程中，平均总奖励的演变情况。这两张平均奖励图确实相当嘈杂，给人的印象是学习算法没有取得稳定的进展。另一个更稳定的指标是策略的估计行动价值函数Q，它提供了agent任何给定状态遵循其策略可以获得多少折现奖励的估计。我们在训练开始前通过运行随机策略收集一组固定的状态，并跟踪这些状态的最大预测Q的平均值。

图中最右边的两幅图显示，平均预测Q的增长比agent获得的平均总奖励要平稳得多，在其他五场比赛中绘制同样的指标也会产生类似的平稳曲线。在训练期间，除了看到预测Q的相对平滑的改善外，我们在任何实验中都没有遇到任何发散问题。这表明，尽管缺乏任何理论上的收敛保证，我们的方法能够以稳定的方式使用强化学习信号和随机梯度下降来训练大型神经网络。



图表 14 论文结果图片

在我实现的Cartpole环境中，可以看到，DQN 的性能在 100 个序列后很快得到提升，最终收敛到策略的最优回报值 200。我们也可以看到，在 DQN 的性能得到提升后，它会持续出现一定程度的震荡，这主要是神经网络过拟合到一些局部经验数据后由argmax运算带来的影响。



图表 15 实验结果图片

# 5.第五部分

## 5.1 DQN优缺点

【DQN的优点】

1.可以应用于高维、连续动作空间，例如ATARI游戏。

2.DQN可以使用经验回放机制，将已有的经验存储下来并随机采样来训练模型，使其更加稳定和数据效率更高。

3.DQN使用了一个目标网络来解决感官误差与价值估计误差问题，提高了算法的收敛稳定性。

4.经典Q-learning更新存在过度依赖当前最大位置（greedy policy）的问题，而DQN采用ε-greedy策略可能会在一定程度上缓解这个问题。

【DQN的缺点】

1.DQN不适合处理持续变化、非静态环境下的任务，因为其基于决策人在可重复状态中操作。如果环境的状态或代理自身已改变，这样的假设将被破坏。

2.DQN难以通过奖励或成本吸引代理探索更多。

3.训练DQN可能需要花费相当长的时间。一般情况下，至少需要数十万个状态转换才能看到表现的显着提高。

## 5.2 未来潜在研究方向

1.改进DQN算法的稳定性。DQN算法存在一些问题，如样本不平衡、目标Q值的更新等，可能导致算法不稳定甚至发生崩溃。未来可以探索更加稳定的算法结构和训练技巧，以提高算法的可靠性。

2.更好地处理高维状态空间。DQN算法常用CNN等卷积神经网络结构来处理像素级别的输入数据，这种方法对于小规模游戏环境有效，但是在大规模游戏环境或者连续控制任务中往往需要更复杂的神经网络结构。未来可以探索更加适应高维状态空间的算法和模型。

3.引入注意力机制和记忆增强方法。由于DQN算法只能处理固定长度的历史状态序列，因此很难捕捉到长期依赖关系。引入注意力机制和记忆增强方法可以允许算法对某些特定的状态进行更加强的关注，并且有效地缓解长期依赖问题。

4.组合多种算法形成混合强化学习系统。DQN虽然在某些任务中取得出色的效果，但是在某些特定的任务中可能并不理想。未来可以考虑将多种强化学习算法组合成混合强化学习系统，以达到更加广泛的应用场景。