1. DRL

原因：

在普通的维度不高，且离散的时候，Q-Learning 算法使用Q-Table 矩阵存储状态

当维度变高，状态和动作空间高维连续的时候使用，Q-Table 存储不了

解决办法：函数拟合。

Q(s,a;θ)≈Q′(s,a)

而深度神经网络可以自动提取复杂特征，因此，面对高维且连续的状态使用深度神经网络最合适不过了。

DRL是将深度学习（DL）与强化学习（RL）结合，直接从高维原始数据学习控制策略。而DQN是DRL的其中一种算法，它要做的就是将卷积神经网络（CNN）和Q-Learning结合起来，CNN的输入是原始图像数据（作为状态State），输出则是每个动作Action对应的价值评估Value Function（Q值）。

1. DL与 RL的结合

Q-Learning：

Q∗(s,a)=Q(s,a)+α(r+γmaxa′Q(s′,a′)−Q(s,a))

DQN的Loss Function为：

L(θ)=E[(TargetQ−Q(s,a;θ))2]

其中 θ 是网络参数，目标为

TargetQ=r+γmaxa′Q(s′,a′;θ)

1. 解决的问题
2. 经验池：经验池的功能主要是解决相关性及非静态分布问题。具体做法是把每个时间步agent与环境交互得到的转移样本 (st,at,rt,st+1) 储存到回放记忆单元，要训练时就随机拿出一些（minibatch）来训练。（其实就是将游戏的过程打成碎片存储，训练时随机抽取就避免了相关性问题）
3. 目标网络：在Nature 2015版本的DQN中提出了这个改进，使用另一个网络（这里称为TargetNet）产生Target Q值。具体地，Q(s,a;θi) 表示当前网络MainNet的输出，用来评估当前状态动作对的值函数；Q(s,a;θ−i) 表示TargetNet的输出，代入上面求 TargetQ 值的公式中得到目标Q值。根据上面的Loss Function更新MainNet的参数，每经过N轮迭代，将MainNet的参数复制给TargetNet。

引入TargetNet后，再一段时间里目标Q值使保持不变的，一定程度降低了当前Q值和目标Q值的相关性，提高了算法稳定性。

1. 流程图



