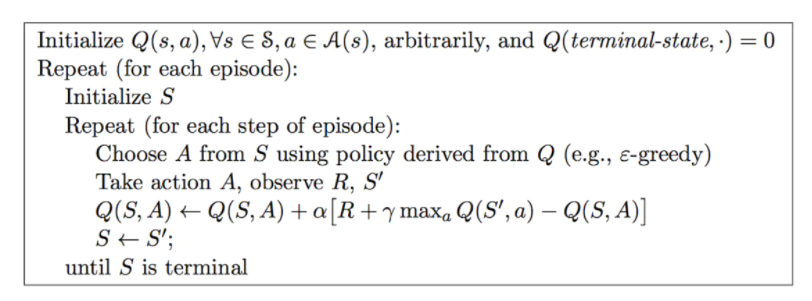
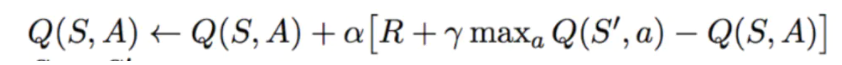
1. DQN产生的背景

Q-learning的算法过程如下图所示：

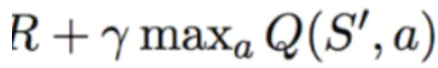


在Q-learning中，维护一张Q值表，表的维数为：状态数S \* 动作数A，表中每个数代表在当前状态S下可以采用动作A可以获得的未来收益的折现和。不断的迭代Q值表使其最终收敛，然后根据Q值表我们就可以在每个状态下选取一个最优策略。

Q值表的更新公式为：



公式中，Q(S,A) 我们可以称做Q估计值，即我们当前估计的Q值，而：

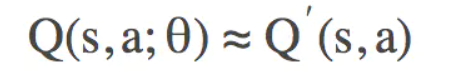


称为Q-target，即我们使用贝尔曼方程加贪心策略认为实际应该得到的奖励，我们的目标就是使我们的Q值不断的接近Q-target值。

在普通的Q-learning中，当状态和动作空间是离散且维数不高时可使用Q-Table储存每个状态动作对的Q值，而当状态和动作空间是高维连续时，使用Q-Table不现实。

**如何将原始的Q-learning转换成深度学习问题？**

将Q-Table的更新问题变成一个函数拟合问题，相近的状态得到相近的输出动作。如下式，通过更新参数 θ 使Q函数逼近最优Q值 。因此，DQN就是要设计一个神经网络结构，通过函数来拟合Q值，即：



DQN整体算法框架与Q-learning相似，而Q-learning是一种off-policy、时间差分的算法。这里首先解释off-policy和时间差分的含义。off-policy是指输出动作的策略（产生样本的策略）与要评估的策略（进行改善的策略）不是同一个策略，例如Q-learning中使用ϵ贪婪策略输出动作，而要评估和改善的策略则是贪婪策略。时间差分方法则是一种采用时间差分目标更新当前动作值函数的方法，Q-learning中TD（时序差分算法）目标为

PV}BAZW`L)TAB$PX~MDPFLX

DQN对Q-learning的修改之处主要体现在以下三点：

* 利用CNN进行值函数拟合

利用CNN进行值函数拟合属于非线性拟合，其中使用一组参数θ来表示动作值函数，在神经网络中θ是每层网络的权重。如果用公式表示，则动作值函数为：Q ( s , a ; θ ) 。

注意，此时更新值函数就是更新参数θ，当网络结构确定下来时，θ就代表值函数。DQN使用的网络结构为3个卷积层和2个全连接层。

* 采用经验回放方法进行强化学习训练

DeepMind团体根据人睡觉时，海马体会将每一天的记忆重放给大脑皮层这一启发式机制，构造出了经验回放的神经网络训练方法，并以此打破强化学习样本的关联性。

训练流程：

1.Agent与环境交互获得样本

2.存入经验池D中：e t → D

3.每隔固定步数，从经验池D DD中随机均匀取出n个样本出来，e k1 ,

ek2 , . . . . . . , ekn 并训练。

* 设置target network单独计算TD目标

与Q-learning相比，此时利用TD目标更新的是参数θ，更新方法是梯度下降，Q-learning梯度下降更新公式为：

6Q9VE5`N{I5LWM%VNQ~6TSU

2.DQN模型关键技术：

* 样本池（Experience Reply）

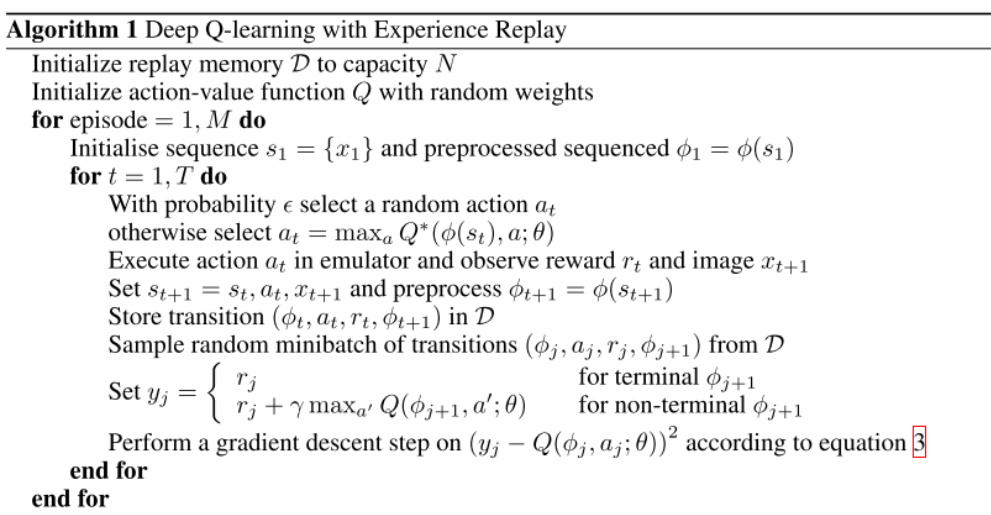
DQN 模型的目标是将强化学习和深层神经网络结合起来，只需要输入原始 RGB 图像并通过 SGD（随机最速下降法） 对样本进行训练便可以输出最优 Q 网络。

DQN 模型中最重要的一个技巧是采用了经验回放机制。在每个时间步长，都会将经验et=(st,at,rt,st+1) 存储在数据集 D={e1,e2,…,eN} 中（其实终止标志 done 也要存进去，表示当前状态是否还有后继状态），并且会保留许多 episode 的经验，即新的 episode 开启时，经验回放池 D 不会重置。打破观察序列的相关性，满足深度网络使用要求；平滑数据分布中的变化，避免训练的振荡，提高采样数据利用效率。

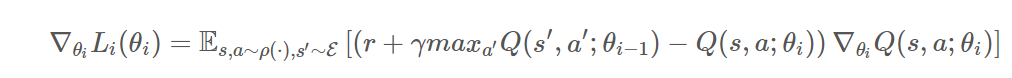
DQN 模型中采用 off-policy，与Q-learning是一样的，目标策略采用贪婪策略，行动策略采用ϵ-贪婪策略。

因为从经验回放池中抽取的历史经验具有不一样的长度（因为状态实际上是一个动作-观测值序列），因此定义函数ϕ来将长度不一致的历史经验输出为固定长度的历史经验表示。等式（3）就是上面的均方误差损失函数。

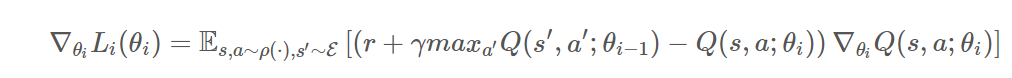
算法伪代码如下：



等式3为：

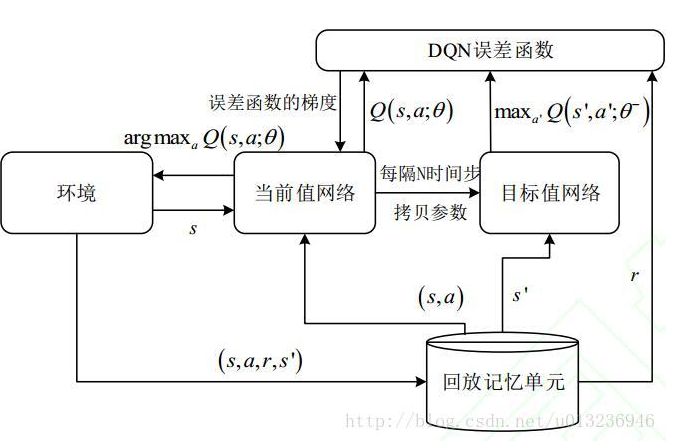


因为样本是从经验回放池D中均匀采样，因此写成下面这种形式：

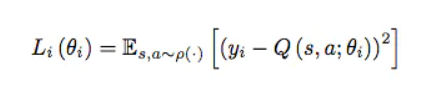


U(D)表示经验回放池D中样本的均匀分布。

* 双网络结构



DQN中的损失函数定义为：



其中，yi也被我们称为q-target值，而后面的Q(s,a)我们称为q-eval值，我们希望q-target和q-eval值越接近越好。

q-target如何计算呢？根据下面的公式：

OU]_}%%CO[MY$~PKD]MXEYL

我们有经验池存储的历史经验，经验池中每一条的结构是(s,a,r,s')，我们的q-target值根据该轮的奖励r以及将s'输入到target-net网络中得到的Q(s',a')的最大值决定。

我们根据状态s'选择动作a'的过程,以及估计Q(s',a')使用的是同一张Q值表，或者说使用的同一个网络参数，这可能导致选择过高的估计值，从而导致过于乐观的值估计。为了避免这种情况的出现，我们可以对选择和衡量进行解耦，从而就有了双Q学习。我们根据一张Q表或者网络参数来选择我们的动作a',再用另一张Q值表活着网络参数来衡量Q(s',a')的值。

DQN除了使用深度卷积网络近似表示当前的值函数之外，还单独使用了另一个网络来产生目标Q值。具体地，Q(s,a)表示当前值网络的输出，用来评估当前状态动作对的值函数。Q(s,a)表示目标值网络的输出,一般采用, Q(s',a')近似表示值函数的优化目标，即目标Q值。当前值网络的参数θ是实时更新的，每经过N轮迭代，将当前值网络的参数复制给目标值网络，通过最小化当前Q值和目标Q值之间的均方误差来更新网络参数。误差函数为：

FBI$VCF1G90_$)X0P}_1NAK

对参数θ求偏导，得到以下梯度：

B4V@A0Q4A_A%@5S%2`(90IV

引入目标值网络后，在一段时间内目标Q值是保持不变的，一定程度上降低了当前Q值和目标Q值之间的相关性，提升了算法的稳定性。  
  DQN将奖赏值和误差项缩小到有限的区间内，保证了Q值和梯度值都处于合理的范围内，提高了算法的稳定性。实验表明，DQN在解决诸Atari 2600游戏等类真实环境的复杂问题时，表现出与人类玩家相媲美的竞技水平，甚至在一些难度较低的非战略性游戏中, DQN的表现超过了有经验的人类玩家。在解决各类基于视觉感知的DRL任务时，DQN使用了同一套网络模型、参数设置和训练算法，这充分说明DQN方法具有很强的适应性和通用性。