老师同学们大家好，这个视频介绍的是DQN算法在俄罗斯方块游戏中的实现。

首先是第一部分：算法流程。DQN网络模型相比于传统的强化学习算法，其独特之处在于可以通过神经网络感知环境，并制定策略、计算Q值。本实验中使用的DQN网络如下，它单独封装成一个类，可以看到它的结构有三层，同时也使用了均匀分布的初始化器，调整了输入输出的分布，并用0填补了bias。它的输入是游戏空间被方块占用的情况，输出是每个动作发生后对应的Q值。

训练开始前需要进行的准备工作如下：首先初始化了俄罗斯方块的一个游戏环境，并实例化了感知它的DQN网络，指定了优化器为Adam。然后是实现了一个回放缓存，还有一些全局变量的初始化。关于环境的初始化还会在每一轮训练开始时执行。

在执行训练的过程中Epsilon-Greedy机制被采用了，他在每次训练时生成一个epsilon，并准备好下一步所有动作和状态，然后预测Q值。接下来有两个分支，一是有epsilon的可能随机选择动作，二是选择预测Q值最大的动作。这使得Agent不局限在最大Q值的少数动作，扩大探索范围。

做出了决策之后就是执行。首先根据决策更新状态和动作变量，然后计算决策做出的效果，包括奖励值（也就是得分）以及这一步后游戏是否结束，最后把这些信息汇总并回放到经验池。

在训练过程中需要更新模型以体现学习效果。为了提高效率和延长学习时间，设置为间隔一定次数进行更新。更新时预测Q值并计算目标Q值，再用两者计算loss，然后执行梯度下降。若到更新频率或者是最高分纪录被打破，会将模型保存到硬盘。

第二部分是本实验的结果分析。奖励值方面在2000个epoch前基本没有起色，随后2000到2500个epoch之间奖励值开始上升，模型开始学习到奖励机制。2700个epoch之后奖励值就收敛了。由于有剧烈的震荡，可能使出现了一定的过拟合。损失值呢在2500到2700个epoch之间收敛，收敛的过程和之后的变化都趋于平缓。

以上就是本次DQN实现的介绍，谢谢大家！