虽然我们Q-learning方法也能训出不错的policy，但是它受到Q-table的存储限制，一方面它在简单模式下的游戏表现依旧不够好，另一方面，它对高维状态空间的可拓展性有限，在更高难度下很难进行有效的学习。所以，我们尝试使用DQN

同时使用Experience replay和Target network来提高稳定性。我们使用权重为θ的f(s, a)来近似Q-value function Q(s, a;θ) (θ为网络的可训练参数)。此时，给定一个state s，神经网络计算所有action对应的Q-value，并选择估计值最高的Q-value action。

在DQN中，每train一定量的episode，我们会从replay buffer D中采样一些转移信息来计算loss并更新网络参数，以提高数据有效性和减少训练样本相关性。而为了稳定性，我们选择定期从行为网络中更新target network权重，来避免权重不断变化引起的不稳定。下面是我们的更新方程y = r + γ max a′ Q(s′, a′; θ−)(1 − d)和最小化Bellman误差方程L(θ) = E(s, a, r, s′, d)∼D(Q(s, a; θ) − y)2].

在具体实验过程中，我们使用了RAM和RGB两种环境来描述agent感知环境状态信息的方式，以及greedy-epsilon来优化agent的动作决策。

首先是RAM环境，下面是我们的一些setup和 reward design（reward design前面Q-learning已经介绍过了）

这是我们RAM环境下训练的agent实测效果

RAM环境的状态信息是直接采用内存中向量信息，维度较低，所以计算更快，但是包含的冗余信息很多，提取到的有效特征很少，所以训练效果依然不够好，agent只知道撞到车之后后退并且进行等车，然后不停的向前走，不够智能。所以我们选择将状态信息作为图像传递进去重训了一个agent。

RGB环境基本沿用了RAM环境的setup，并且直接使用了环境给的Reward

我们在状态信息处理时，因为环境状态表示维度更高，为了降低复杂度并保证计算效率，我们选择将RGB图像转换成灰度图，并调整为84×84像素，使用卷积神经网络从图像中提取特征进行输出，将环境空间维度转换成agent的动作维度。

因为训练时间较长，我们分为三段进行训练，首先调小replay buffer，通过训练较少的episode来达到Q-learning和RAM环境下的agent水平，然后将保存下来的ckpt作为model重新训练，让agent以较高的探索率探索优化策略，最后回调buffer，做最后训练，让agent表现收敛到理想值。

这是我们RGB环境下训练的agent实测效果

因为在RGB环境中，我们显式的进行了特征选择，所以提取到的有效特征较多。在两种难度下agent都有明显的等车和躲车动作。因为mode1难度较高，要学习到持续的躲车行为较难。即使如此，表现已经接近，或略优于human表现。

我们使用了Gym和Atari作为我们的游戏环境，借助提供的函数接口向agent传递一些状态信息并进行训练。