基于CNN与DQN的Flappybird深度学习实践

组长：周家豪

组员：吴祖峰

组员：陈天翼

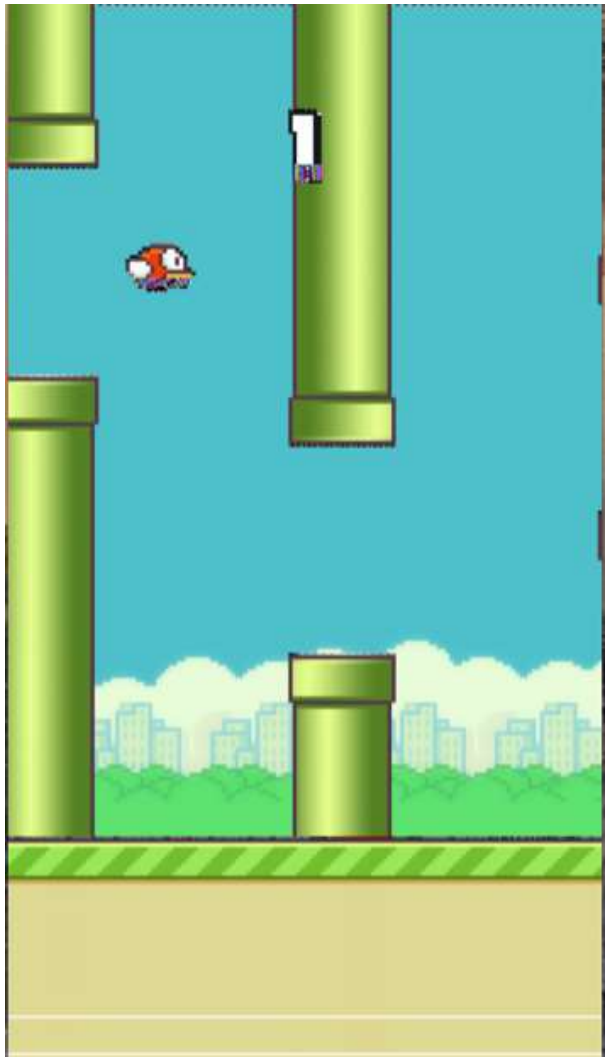
组员：董腾然

项目概述

三

项目概述

该项目目的是开发一个深层神经网络模型，具体来说，是利用图像中的不同对象训练卷积神经网络，进行基于游戏画面场景状态分析进行图像识别分类。从原始像素中学习游戏的特性，并决定采取相应行动，本质上是一个对游戏场景中特定状态的模式识别过程，在此设计了一个强化学习系统，通过自主学习来玩这款游戏。



亮点：将深度学习模型与强化学习结合在一起从而成功地直接从高维的输入学习控制策略

一

算法介绍

三

什么是CNN?

- 卷积神经网络（Convolutional Neural Networks, CNN）是一类包含卷积计算且具有深度结构的前馈神经网络，是深度学习（deep learning）的代表算法之一 [1-2] 。卷积神经网络具有表征学习（representation learning）能力，能够按其阶层结构对输入信息进行平移不变分类（shift-invariant classification），因此也被称为“平移不变人工神经网络（Shift-Invariant Artificial Neural Networks, SIANN）” 。
- 卷积的主要目的是使信号增强，同时降低噪音。对图像用卷积核进性运算，实际上是一个滤波过程。每个卷积核都是一种特征提取方式，就像一个筛子，将图像中符合条件的部分提取出来。

什么是DQN?

- DQN(Deep Q-Network),即结合了深层神经网络的强化学习的深度Q网络（DQN）模型
- 它是将深度学习deeplearning与强化学习reinforcementlearning相结合，实现了从感知到动作的端到端的一个卷积神经网络
- 使用Q学习的变体进行训练，其输入是原始像素，其输出是估计未来奖励的值函数

DQN伪代码

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-}=\theta$

For episode = 1, M **do**

  Initialize sequence $s_1=\{x_1\}$ and preprocessed sequence $\phi_1=\phi(s_1)$

  For t = 1,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t=\text{argmax}_a Q(\phi(s_t),a;\theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1}=s_t,a_t,x_{t+1}$ and preprocess $\phi_{t+1}=\phi(s_{t+1})$

    Store transition $(\phi_t,a_t,r_t,\phi_{t+1})$ in D

    Sample random minibatch of transitions $(\phi_j,a_j,r_j,\phi_{j+1})$ from D

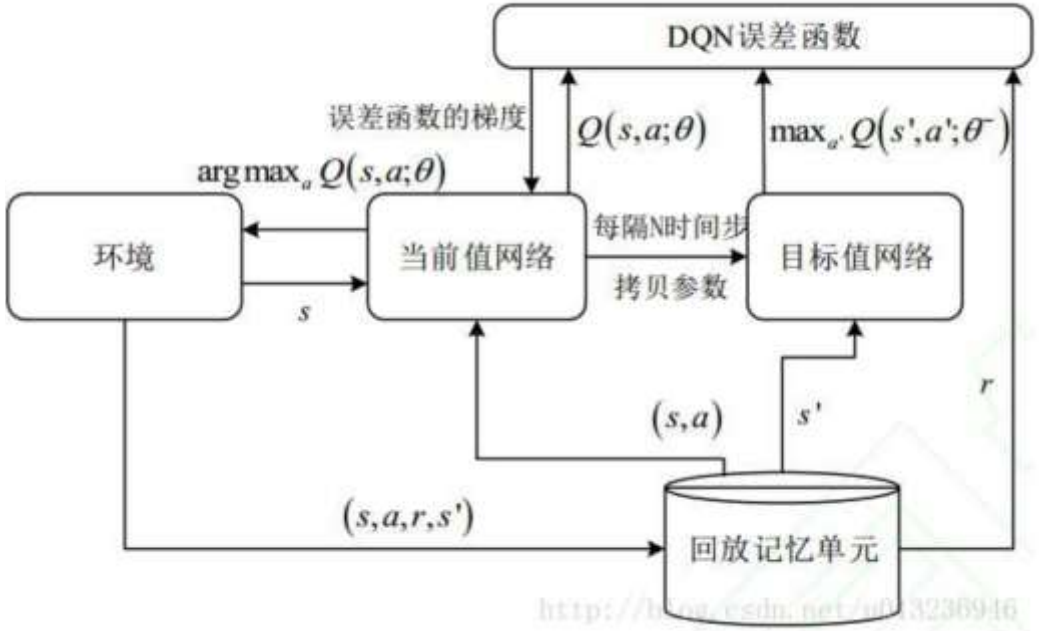
    Set $y_j=\begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j+\gamma \max_{a'} \hat{Q}(\phi_{j+1},a';\theta^{-}) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j-Q(\phi_j,a_j;\theta)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q}=Q$

  End For

End For



实现方法

三

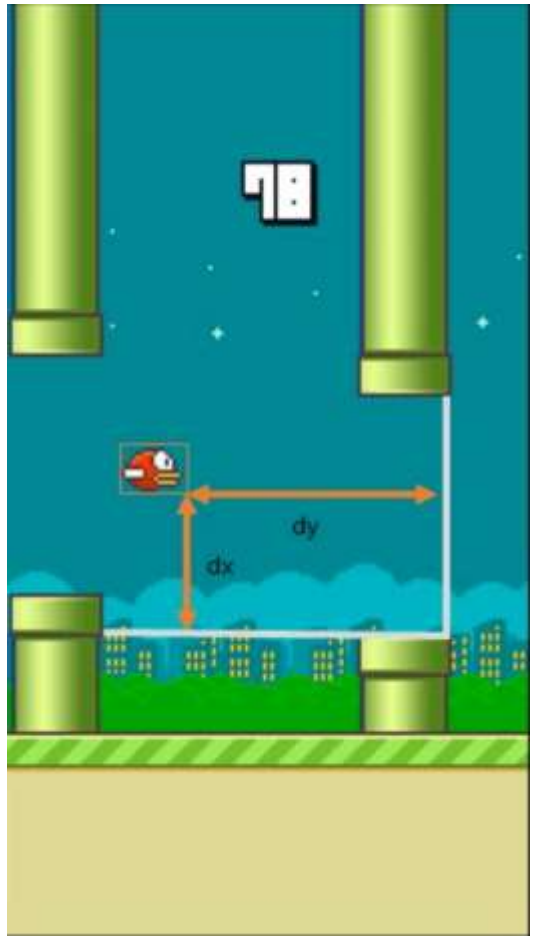
实现方法

- 该项目是通过训练一个深度卷积神经网络模型（深度Q学习网络）进行特定游戏状态下图像的识别与分类。
- 卷积神经网络的任务是提取游戏运行的图像，并输出从可采取的操作集合中提取的必要执行动作。
- 强化学习的任务是根据执行游戏并基于所观察到的奖励来评价一个给定状态下的动作，以此来进行模型训练。

实现方法

让小鸟学习怎么飞是一个强化学习的过程，强化学习中有状态、动作、奖赏三个要素，小鸟根据状态采取动作，获得奖赏后再去改进这些动作，使下次再到相同的状态，小鸟能做出更优的动作

- **状态的选择：** 小鸟到下一根下侧管子的水平距离和垂直距离差
- **动作的选择：** 0表示什么都不做，1表示向上飞一下
- **奖励的选择：** 活着的时候 每一帧给予0.1，死亡，给予-1 成功经过一个水管，则给予1
- https://blog.csdn.net/pihe7623/article/details/80234263

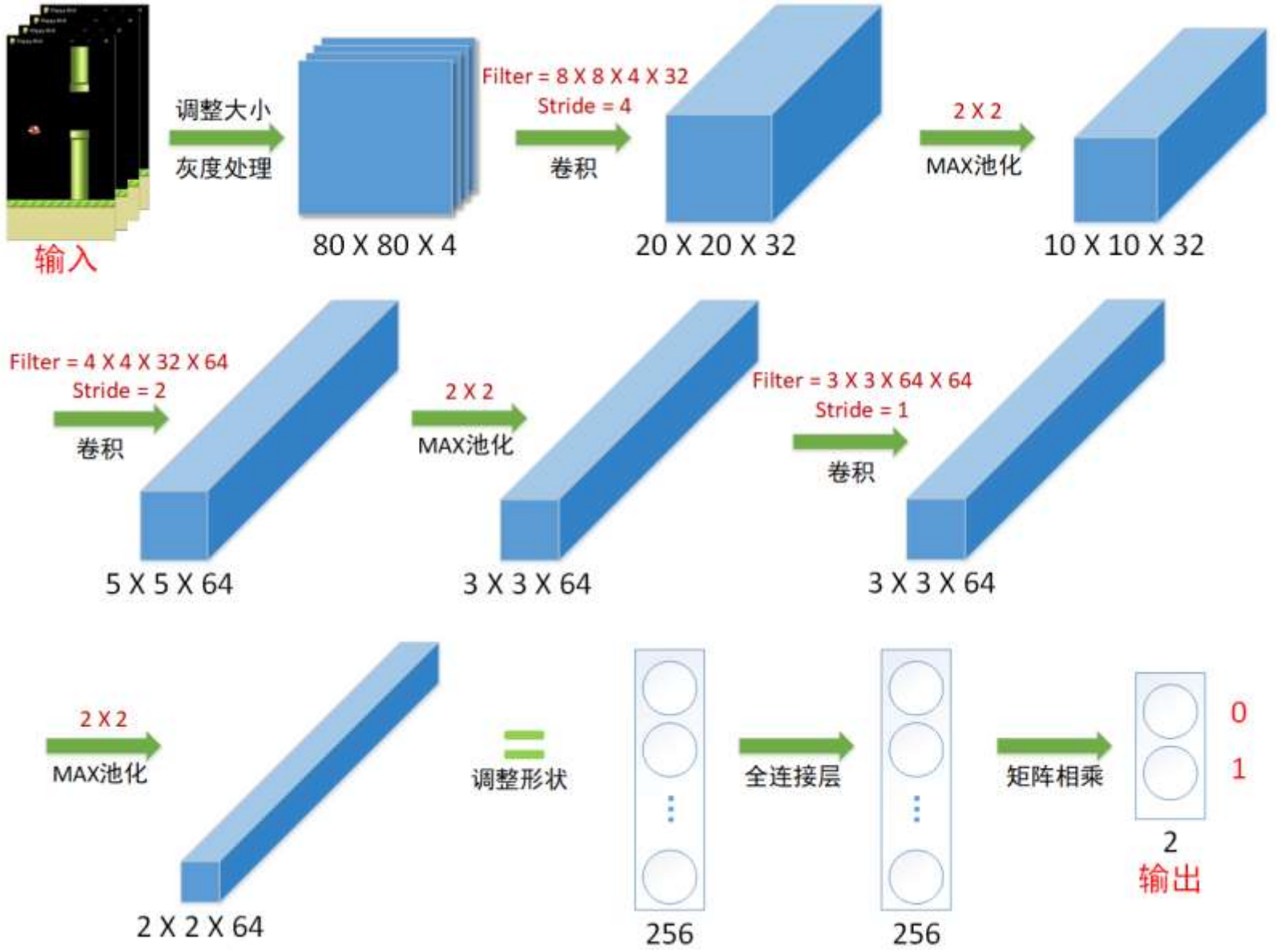


算法架构

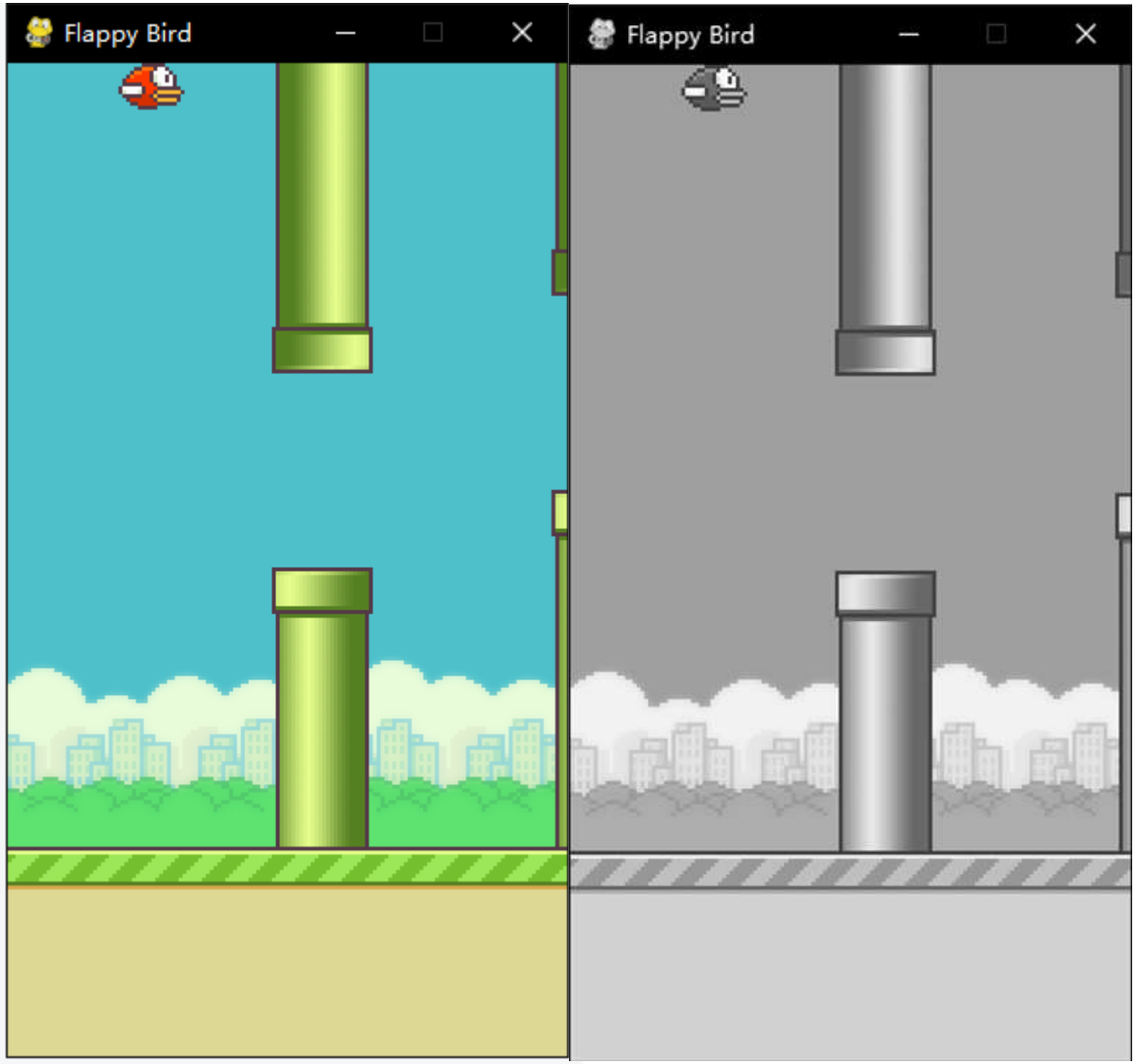
三

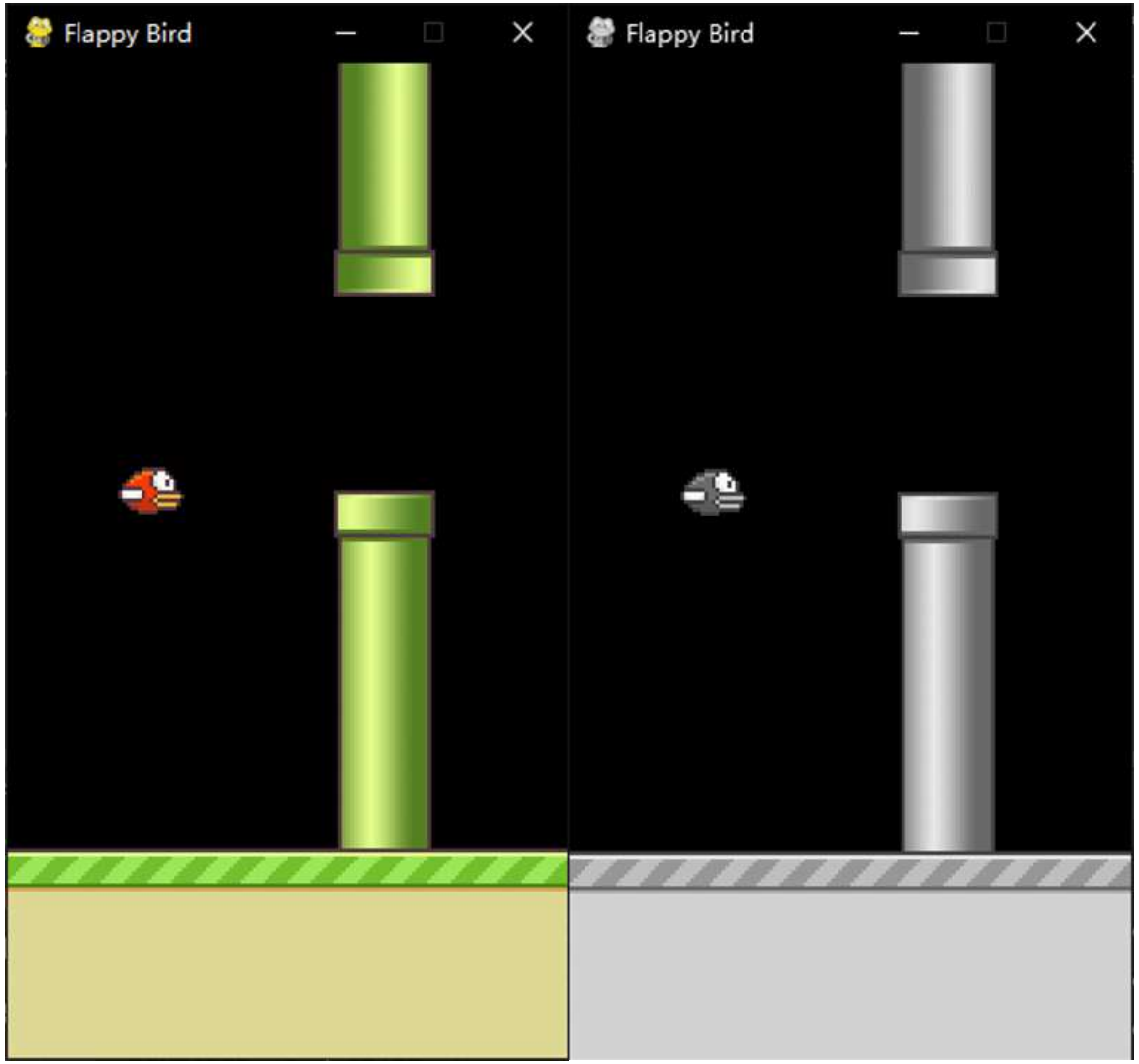
CNN网络架构

- 将图像转换为灰度
- 将图像大小调整为80x80
- 堆叠最后4帧以产生用于网络的80x80x4输入阵列



有无背景图片灰度处理对比图





DQN策略

$$Q(S,A) \leftarrow (1-\alpha) * Q(S,A) + \alpha * [R + \gamma * max_a Q(S',a)]$$

其中α为学习率（learning rate），γ为折扣因子（discount factor）。根据公式可以看出，学习速率α越大，保留之前训练的效果就越少。折扣因子γ越大，max_aQ(S', a)所起到的作用就越大。

小鸟在对状态进行更新时，会考虑到眼前利益（R），和记忆中的利益（max_aQ(S', a)）

max_aQ(S', a)指的便是记忆中的利益。它是指小鸟记忆里下一个状态S'的动作中效用值的最大值。如果小鸟之前在下一个状态S'的某个动作上吃过甜头（选择了某个动作之后获得了50的奖赏），那么它就更希望提早地得知这个消息，以便下回在状态S可以通过选择正确的动作继续进入这个吃甜头的状态S'

可以看出，γ越大，小鸟就会越重视以往经验，越小，小鸟越重视眼前利益（R）。

函数介绍

三

函数介绍

- __init__(self)　　#初始化函数，定义了小鸟初始位置和管道出现位置
- frame_step(self, input_actions)　　#用于调整帧数，帧数会随分数增加逐渐提高以增加游戏难度
- getRandomPipe()　　#随机取上下管道的距离
- showScore(score)　　#显示分数
- checkCrash(player, upperPipes, lowerPipes)　　#碰撞检测函数
- pixelCollision(rect1, rect2, hitmask1, hitmask2)　　#检测两物体是否接触

函数介绍

- weight_variable(shape)　　#定义一个函数用来初始化所有的权值w
- bias_variable(shape):　　#用于初始化所有的偏置项
- conv2d(x, W, stride)　　#用于构建卷积层
- max_pool_2x2(x)　　#用于构建池化层
- createNetwork()　　#构建神经网络
- trainNetwork(s, readout, h_fc1, sess)　#训练函数