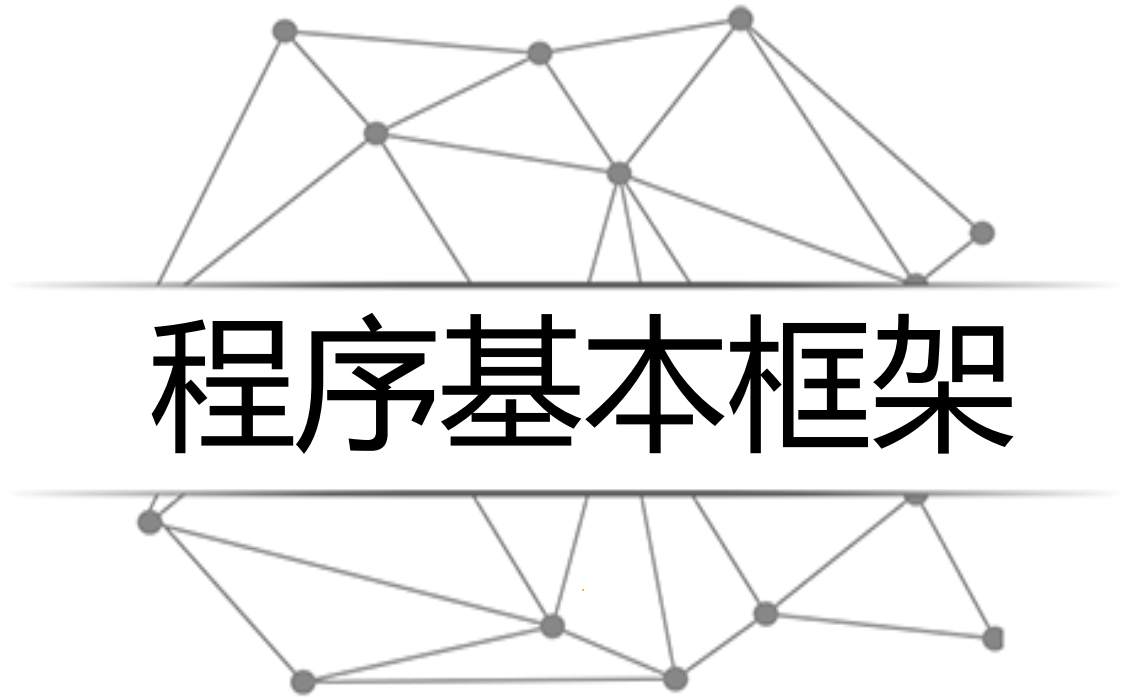
Flappy Bird自主学习程序基本框架

ML26

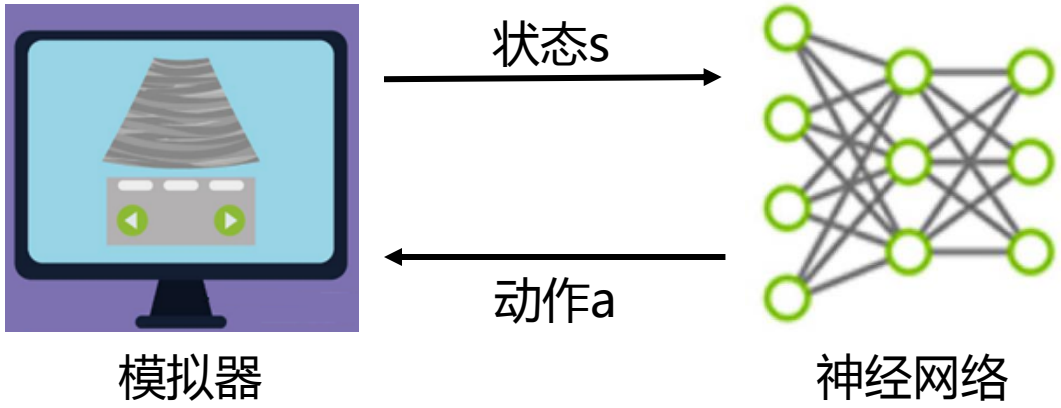
礼欣
www.python123.org



程序基本框架

程序与模拟器交互

训练过程也就是神经网络（agent）不断与游戏模拟器（Environment）进行交互，通过模拟器获得状态，给出动作，改变模拟器中的状态，获得反馈，依据反馈更新策略的过程。



训练过程

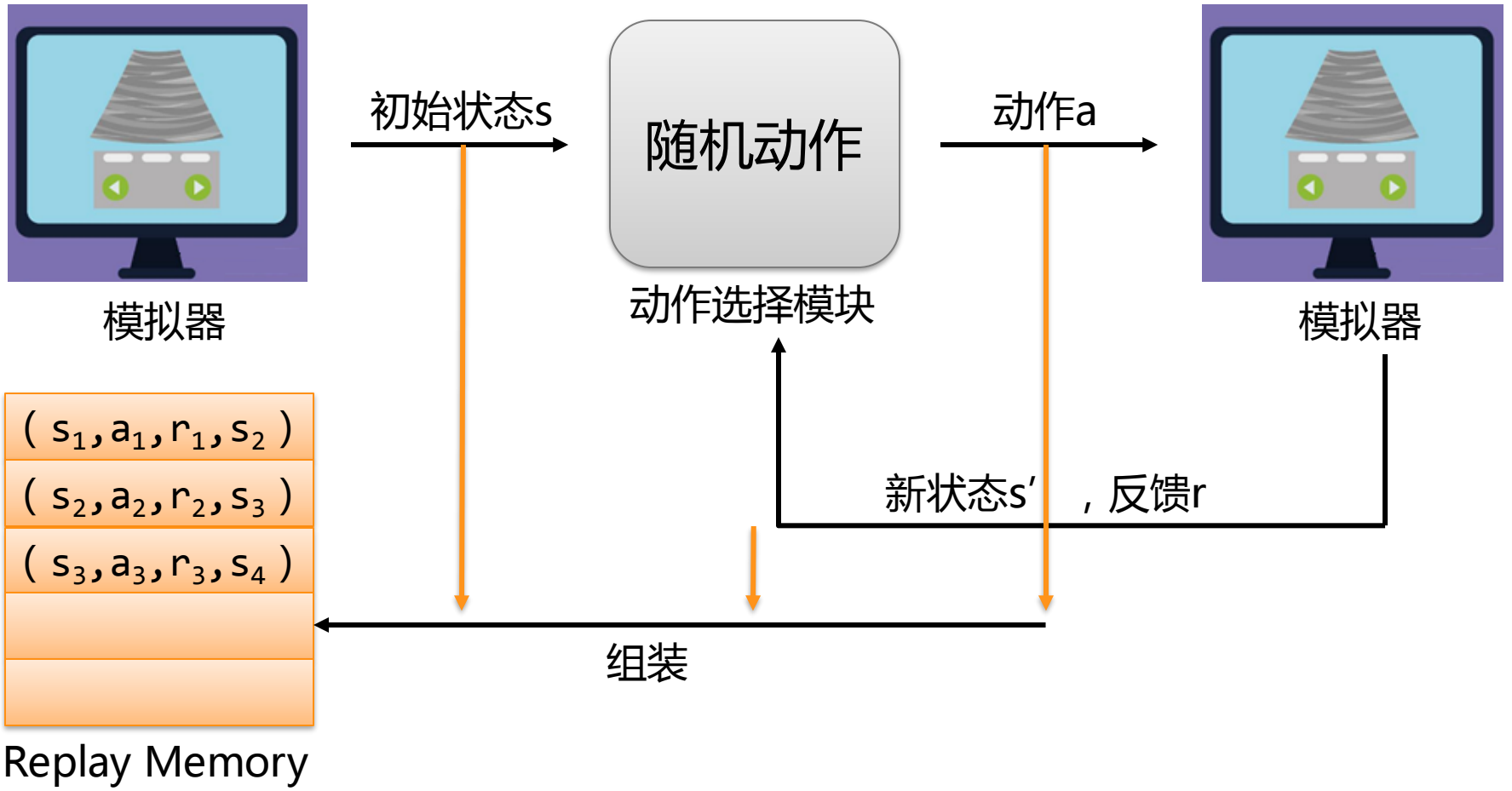
训练过程过程主要分为以下三个阶段：

1.观察期（OBSERVE）：程序与模拟器进行交互，随机给出动作，获取模拟器中的状态，将状态转移过程存放在D（Replay Memory）中；

2.探索期（EXPLORE）：程序与模拟器交互的过程中，依据Replay Memory中存储的历史信息更新网络参数，并随训练过程降低随机探索率ε;

3.训练期（TRAIN）：ε已经很小，不再发生改变，网络参数随着训练过程不断趋于稳定。

整体框架—观察期


初始状态s
随机动作
动作a
模拟器
动作选择模块
模拟器
新状态s′ ，反馈r
组装
(s_1,a_1,r_1,s_2)
(s_2,a_2,r_2,s_3)
(s_3,a_3,r_3,s_4)
Replay Memory

整体框架—观察期

1.打开游戏模拟器，不执行跳跃动作，获取游戏的初始状态

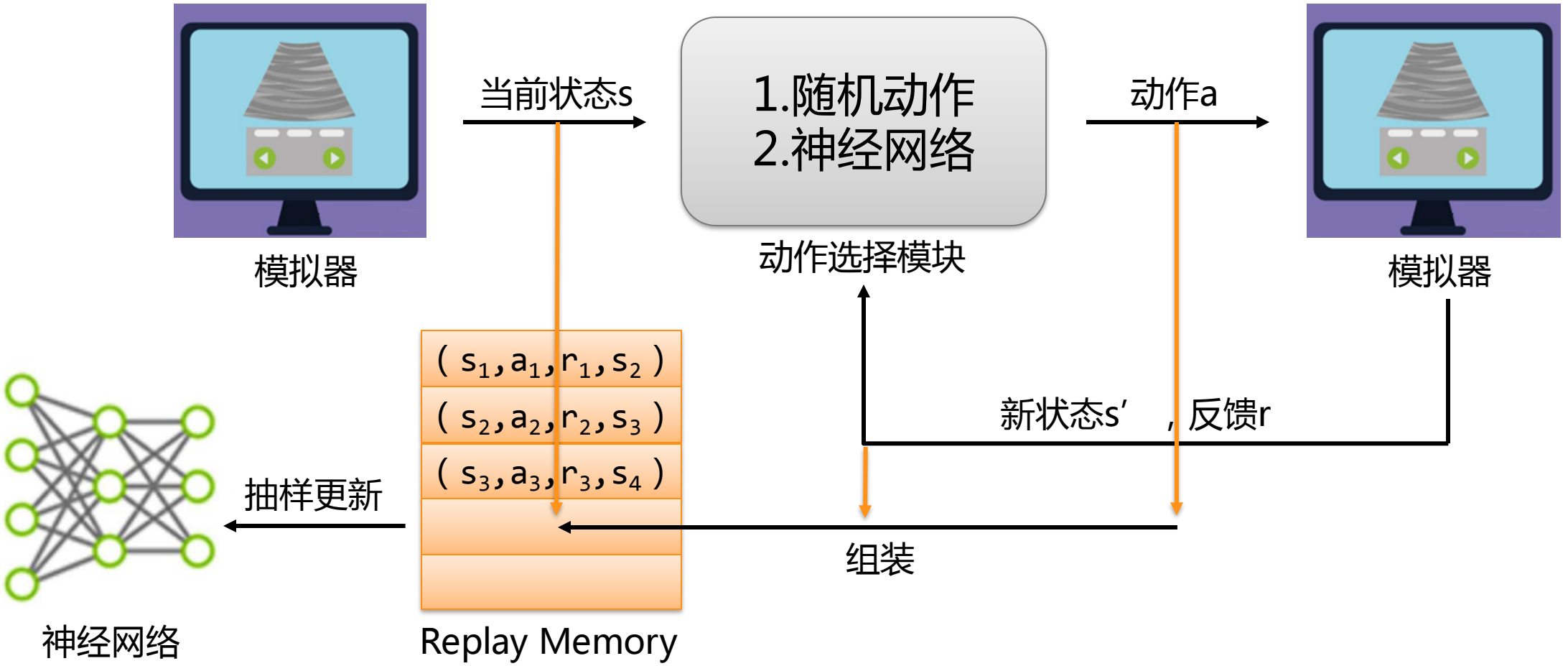
2.根据ε贪心策略获得一个动作（由于神经网络参数也是随机初始化的，在本阶段参数也不会进行更新，所以统称为随机动作），并根据迭代次数减小ε的大小

3.由模拟器执行选择的动作，能够返回新的状态和反馈奖励

4.将上一状态s，动作a，新状态s ‘，反馈r组装成（s，a，s’ ，r）放进Replay Memory中用作以后的参数更新

5.根据新的状态s ‘，根据ε贪心策略选择下一步执行的动作，周而复始，直至迭代次数到达探索期

整体框架—探索期


模拟器
当前状态s
1.随机动作
2.神经网络
动作选择模块
动作a
模拟器
新状态s’ ，反馈r
组装
(s_1,a_1,r_1,s_2)
(s_2,a_2,r_2,s_3)
(s_3,a_3,r_3,s_4)
Replay Memory
抽样更新
神经网络

整体框架——探索期

探索期与观察期的唯一区别在于会根据抽样对网络参数进行更新。

1.迭代次数达到一定数目，进入探索期，根据当前状态s，使用ε贪心策略选择一个动作（可以是随机动作或者由神经网络选择动作），并根据迭代次数减小ε的值

2.由模拟器执行选择的动作，能够返回新的状态和反馈奖励

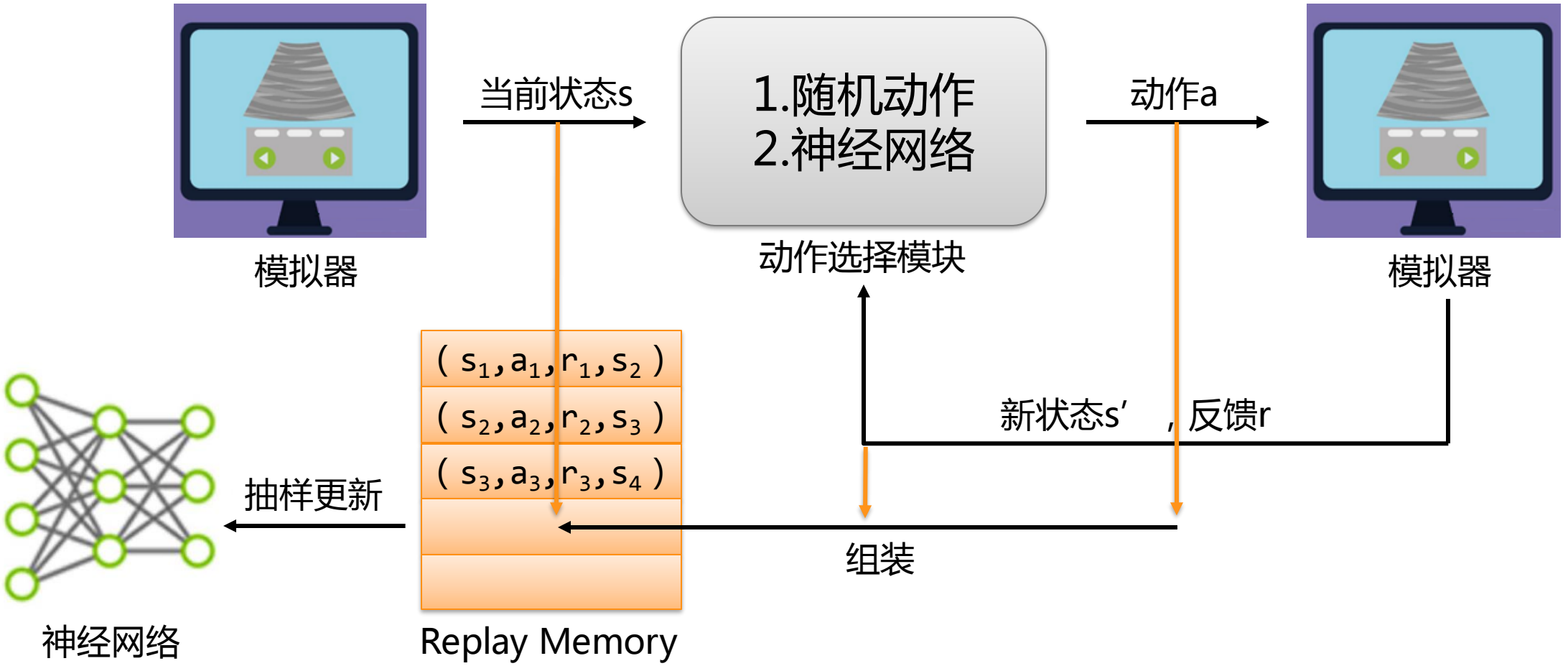
3.将上一状态s，动作a，新状态s ‘，反馈r组装成（s，a，s’ ，r）放进Replay Memory中用作参数更新

4.从Replay Memory中抽取一定量的样本，对神经网络的参数进行更新

5.根据新的状态s ‘，根据ε贪心策略选择下一步执行的动作，周而复始，直至迭代次数到达训练期

整体框架—训练期

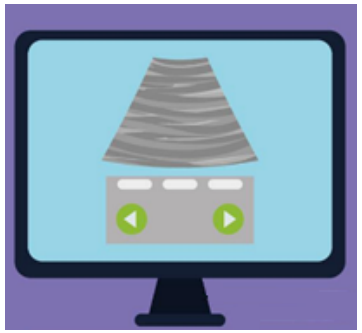
迭代次数达到一定数目，进入训练期，本阶段跟探索期的过程相同，只是在迭代过程中不再修改ε的值



模拟器

游戏模拟器：使用Python的Pygame模块完成的Flappy Bird游戏程序，为了配合训练过程，在原有的游戏程序基础上进行了修改。参考以下网址查看游戏源码：

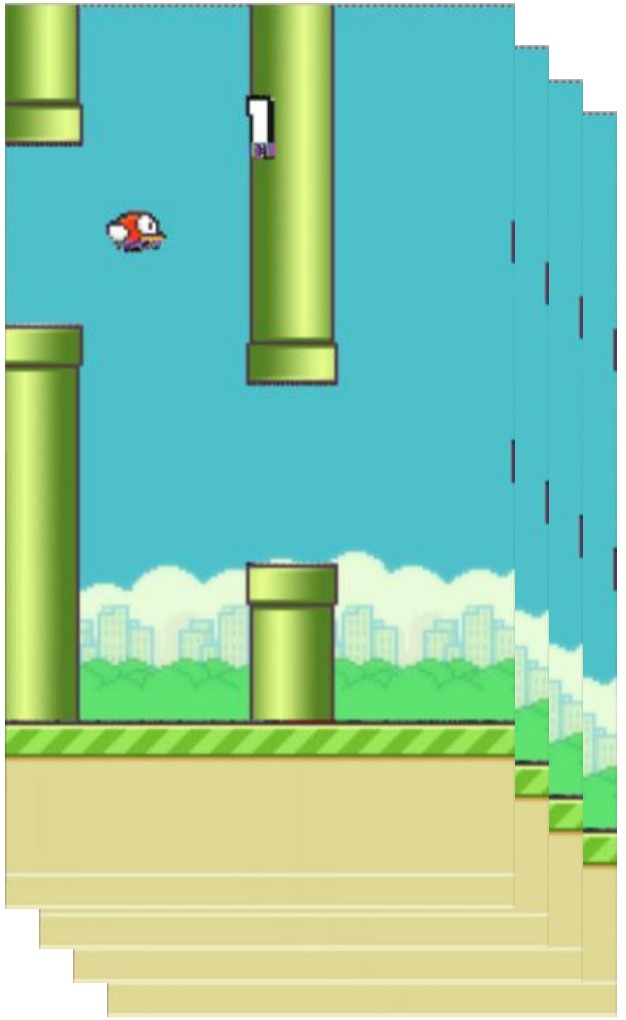
链接：https://github.com/sourabhv/FlapPyBird



模拟器

模拟器

- 图示通过模拟器获取游戏的画面。
- 训练过程中使用连续4帧图像作为一个状态s，用于神经网络的输入。



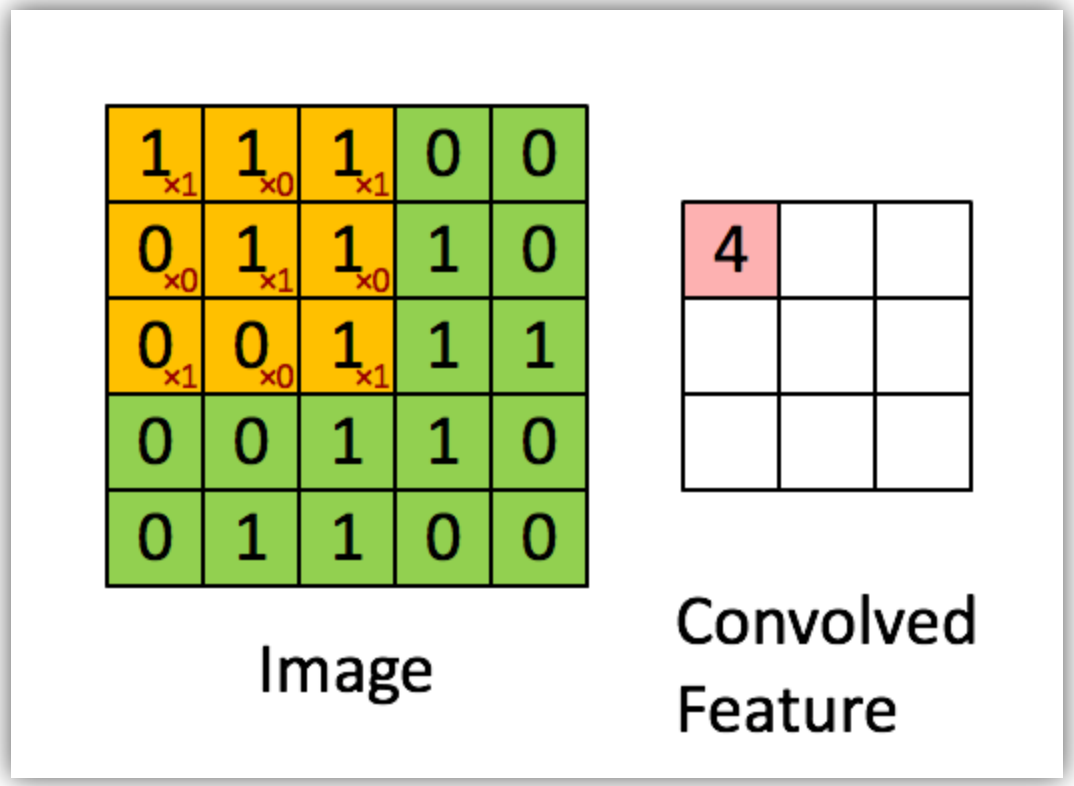
动作选择模块

动作选择模块：为ε贪心策略的简单应用，以概率ε随机从动作空间A中选择动作，以1-ε概率依靠神经网络的输出选择动作：

$$\boldsymbol{\pi(s,a)} = \begin{cases} \boldsymbol{argmax_a Q(s,a)} & \text{以概率} \boldsymbol{1-\varepsilon} \\ \text{随机从} \boldsymbol{A} \text{中选取动作} & \text{以概率} \boldsymbol{\varepsilon} \end{cases}$$

深度神经网络-CNN

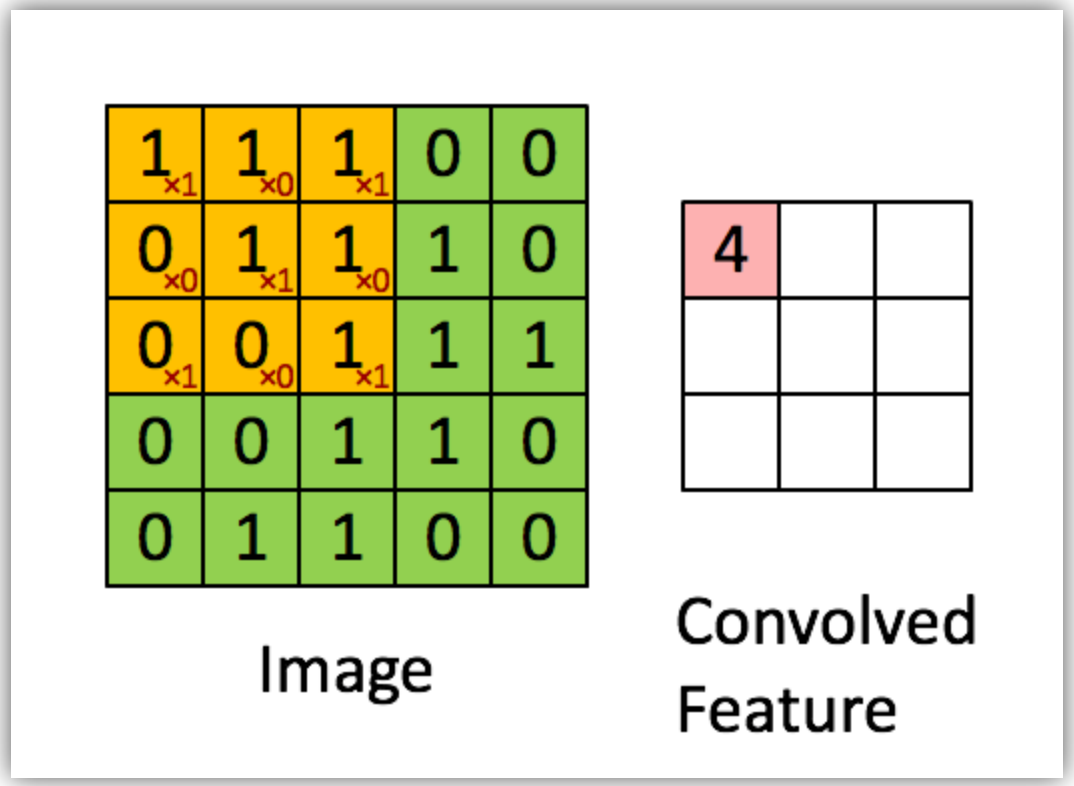
- DQN：用卷积神经网络对游戏画面进行特征提取，这个步骤可以理解为对状态的提取。
- 卷积神经网络(CNN)：右侧展示卷积操作。



CNN-卷积核

卷积核：这里的卷积核指的就是移动中3*3大小的矩阵。

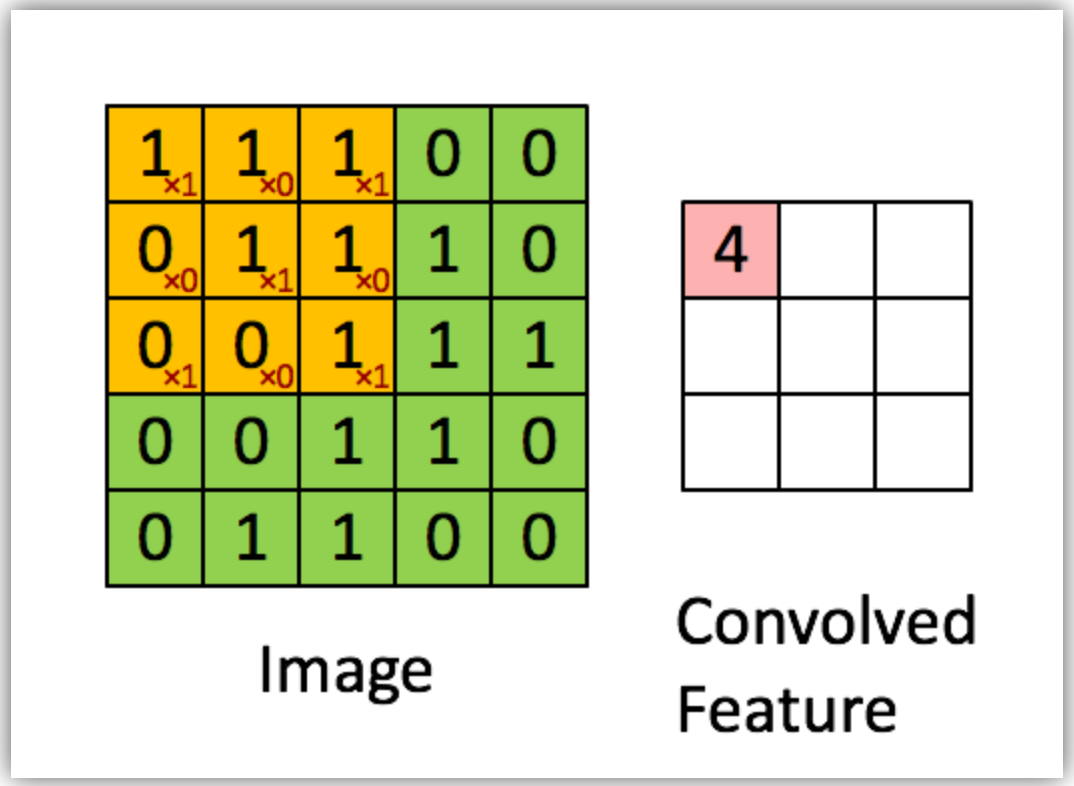
1	0	1
0	1	0
1	0	1



CNN-卷积操作

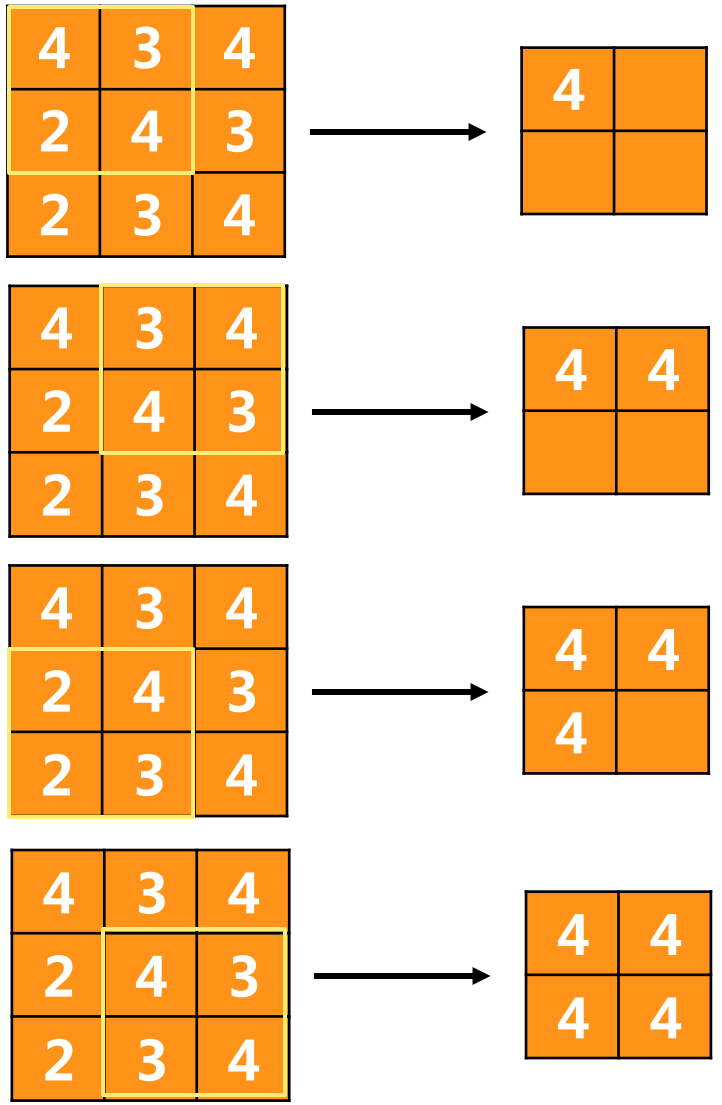
卷积操作：使用卷积核与数据进行对应位置的乘积并加和，不断移动卷积核生成卷积后的特征。

1	0	1
0	1	0
1	0	1



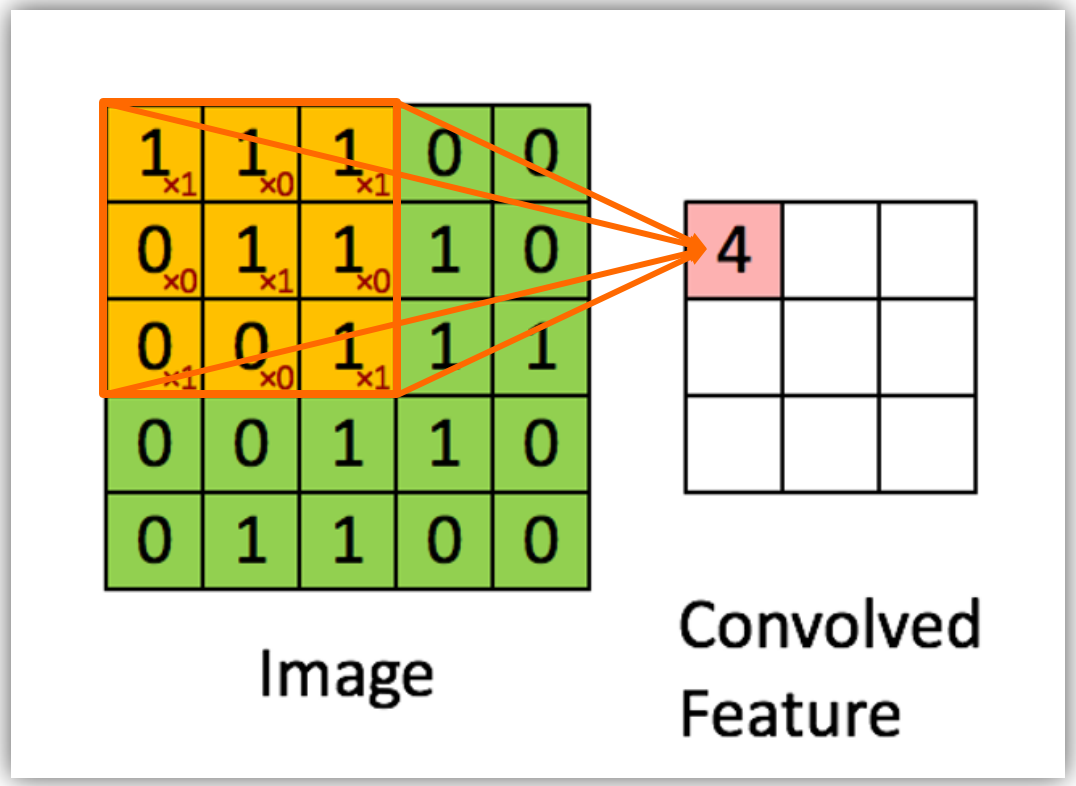
CNN-池化操作

池化操作：对卷积的结果进行操作。最常用的是最大池化操作，即从卷积结果中挑出最大值，如选择一个2*2大小的池化窗口（操作如图示）：

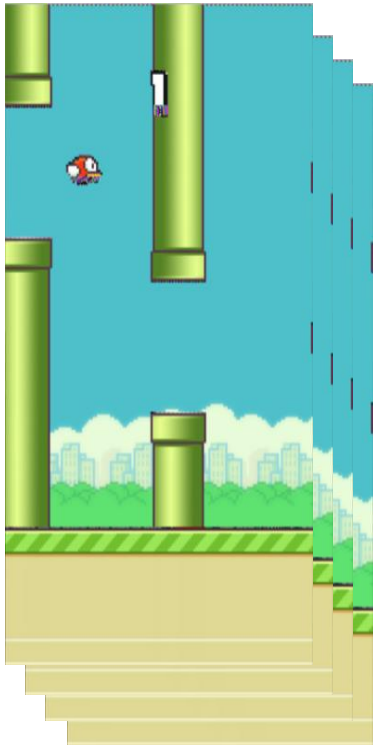


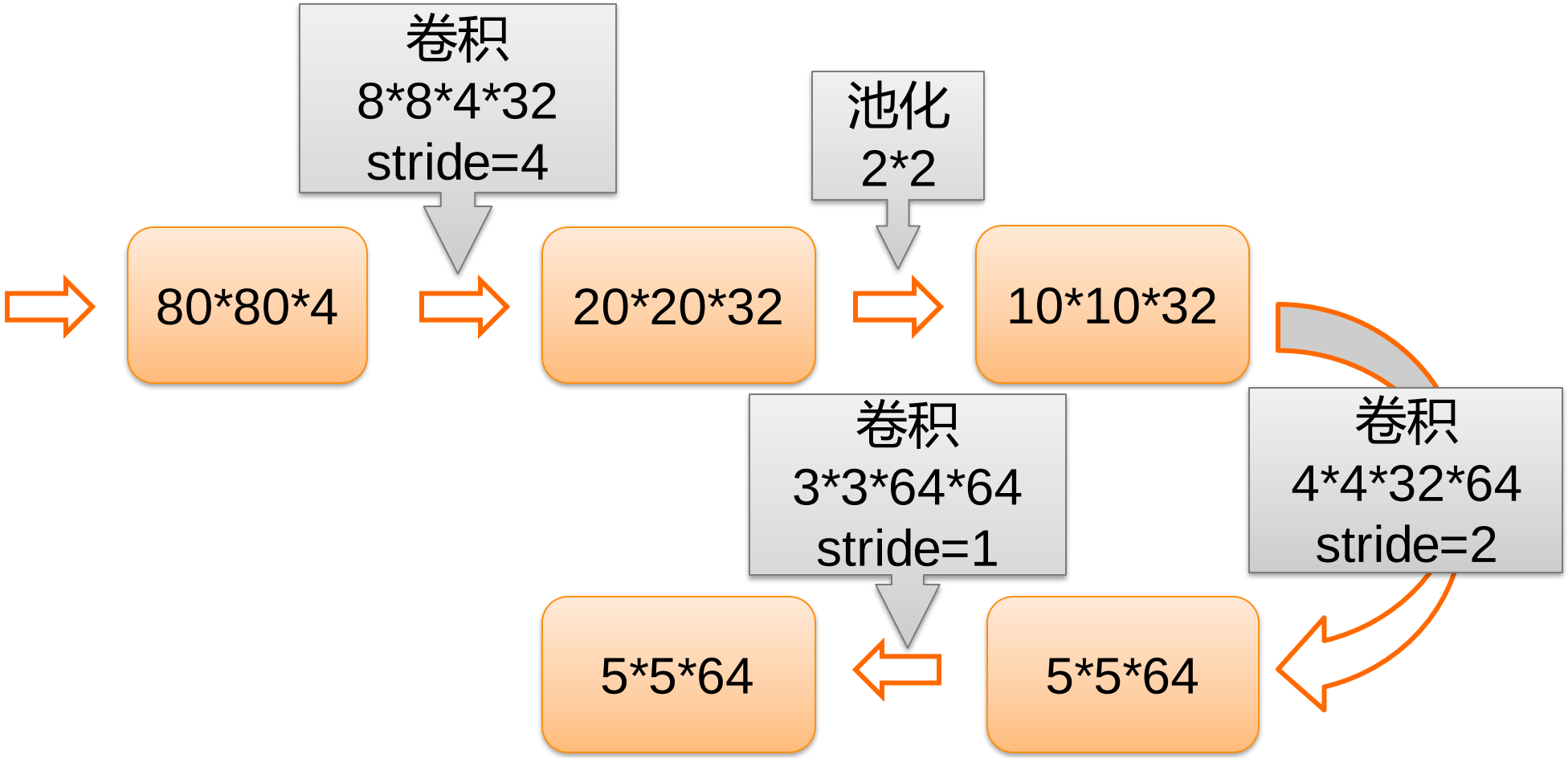
卷积神经网络

卷积神经网络：把Image矩阵中的每个元素当做一个神经元，那么卷积核就相当于输入神经元和输出神经元之间的链接权重，由此构建而成的网络被称作卷积神经网络。



Flappy Bird-深度神经网络



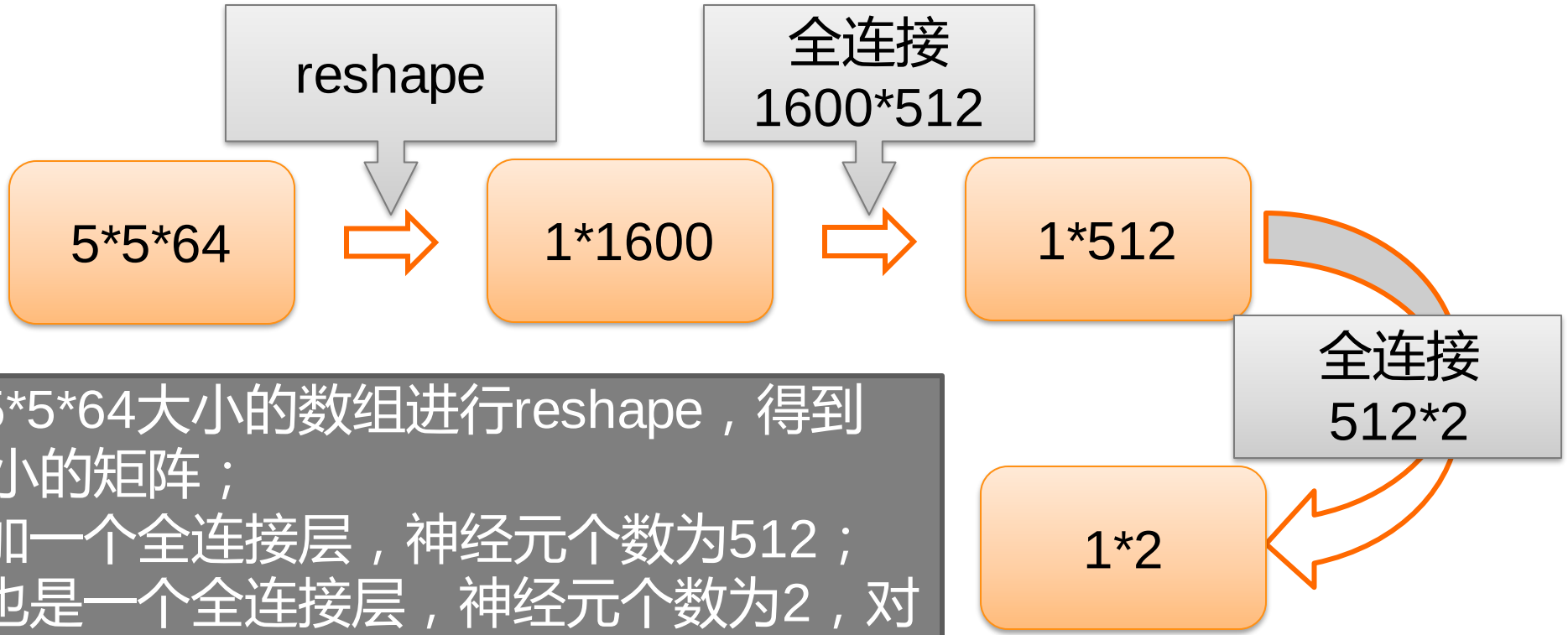


本实验中使用的深度神经网络结构就是多个卷积操作和池化操作的累加。

Flappy Bird-深度神经网络

- 对采集的4张原始图像进行预处理，得到80*80*4大小的矩阵；
- 使用32个8*8*4大小步长4的卷积核对以上矩阵进行卷积，得到20*20*32大小的矩阵；注：在tensorflow中使用4维向量表示卷积核[输入通道数，高度，宽度，输出通道数]，对应于上面的[4,8,8,32]，可以理解为32个8*8*4大小的卷积核；
- 对以上矩阵进行不重叠的池化操作，池化窗口为2*2大小，步长为2，得到10*10*32大小的矩阵；
- 使用64个4*4*32大小步长为2的卷积核对以上矩阵进行卷积，得到5*5*64的矩阵；
- 使用64个3*3*64大小步长为1的卷积核对以上矩阵进行卷积，得到5*5*64的矩阵；

Flappy Bird-深度神经网络



- 将输出的5*5*64大小的数组进行reshape，得到1*1600大小的矩阵；
- 在之后添加一个全连接层，神经元个数为512；
- 最后一层也是一个全连接层，神经元个数为2，对应的是就是两个动作的动作值函数；

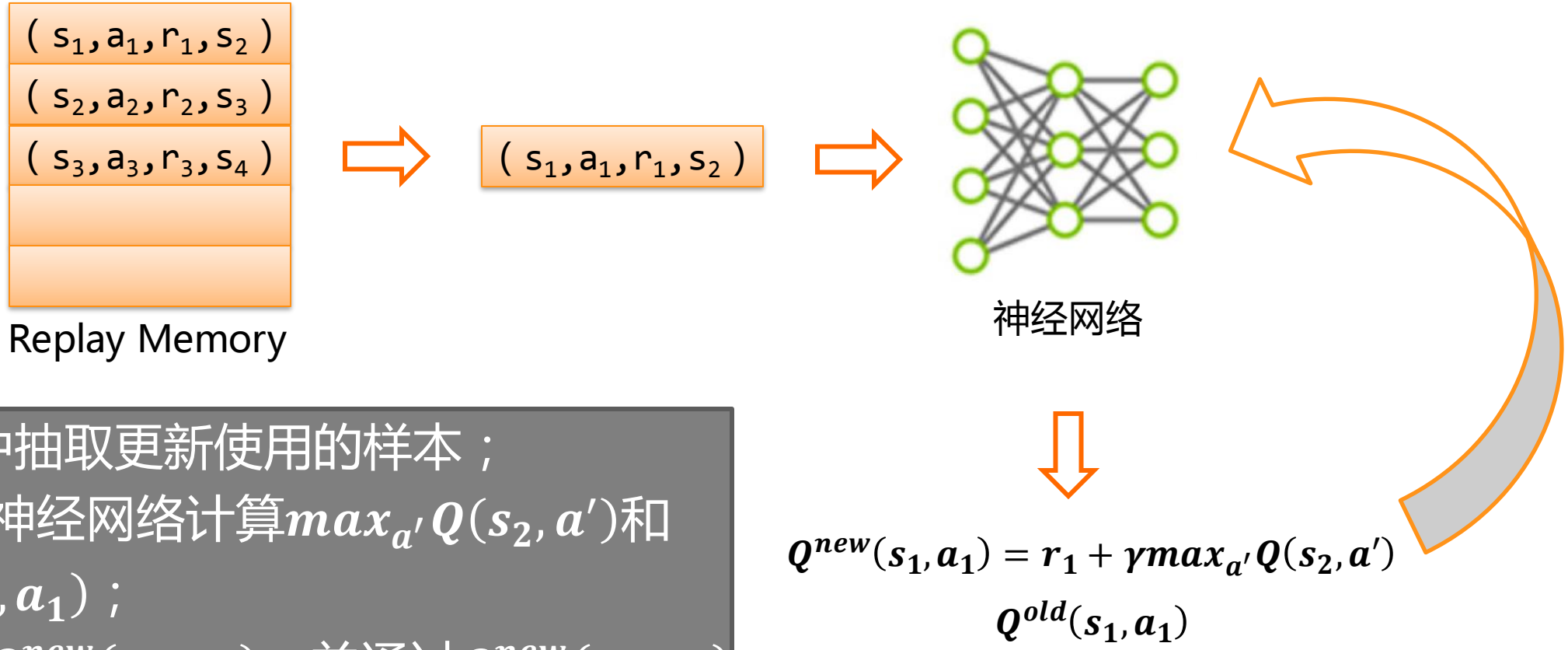
输出分别对应于两个动作，即不做操作和跳跃的状态动作值函数。

Flappy Bird-深度神经网络

通过获得输入s，神经网络就能够：

- 输出Q(s,a1)和Q(s,a2)比较两个值的大小，就能够评判采用动作a1和a2的优劣，从而选择要采取的动作
- 在选择并执行完采用的动作后，模拟器会更新状态并返回回报值，然后将这个状态转移过程存储进D，进行采样更新网络参数。

网络参数更新



1.从D中抽取更新使用的样本；
2.利用神经网络计算$max_{a'} Q(s_2, a')$和$Q^{old}(s_1, a_1)$；
3.计算$Q^{new}(s_1, a_1)$，并通过$Q^{new}(s_1, a_1)$和$Q^{old}(s_1, a_1)$差值更新网络参数。