理解基础:

- 增强学习基本知识
- 深度学习 特别是卷积神经网络的基本知识

创新点：第一个将深度学习模型与增强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

具体是将卷积神经网络和Q Learning结合在一起。卷积神经网络的输入是原始图像数据（作为状态）输则为每个动作对应的价值Value Function来估计未来的反馈Reward

实验成果：使用同一个网络学习玩Atari 2600 游戏，在测试的7个游戏中6个超过了以往的方法并且好几个超过人类的水平。

在这篇文章中，还只是测试7个游戏，到了Nature的文章则测试了更多的游戏，并且取得了更好的效果

优点:

- 算法具备通用性，一样的网络可以学习不同的游戏（当然，游戏具有相似性）
- 采用End-to-End的训练方式，无需人工提取Feature（比如游戏中敌人的位置等等）
- 通过不断的测试训练，可以实时生成无尽的样本用于有监督训练（Supervised Learning）

缺点:

- 由于输入的状态是短时的，所以只适用于处理只需短时记忆的问题，无法处理需要长时间经验的题。（比如玩超级玛丽）
- 使用CNN来训练不一定能够收敛，需要对网络的参数进行精良的设置才行。

改进办法:

- 使用LSTM 来增强记忆性?
- 改进Q-Learning的算法提高网络收敛能力。

详细分析

1 前言介绍 Introduction

提出问题:

直接从高维的输入（比如视觉或听觉）来学习一个控制策略 是 RL增强学习的长期挑战。个人理解：这问题是人工智能抽象出来的极其重要的子问题，因为人类都是通过视觉听觉触觉等感觉然后来学习一项能，比如玩游戏，打篮球，洗碗等等。 解决这个问题的意义在于机器人不一定可以具有自我意识，但可以实现 **机器人彻底代替重复性劳动** 的愿景。

以往的解决办法:

- 人工提取特征（比如物体的位置）
- 使用线性的value function或者policy策略来表征

性能的好坏主要取决于特征提取的好坏

Deep Learning 带来的机会

当前，深度学习已经在视觉，语音等领域取得突破性进展，根本的方法就是通过神经网络自动提取复杂征。所以，很自然的我们会考虑一个问题:
增强学习能否收益于深度学习

答案当然是**YES**

从RL看结合Deep Learning的困难之处

- 深度学习的成功依赖于大量的有标签的样本，从而进行有监督学习。而增强学习只有一个reward信值，并且这个值还常常带有噪声，延迟，并且是稀少的（sparse），理解是不可能每个state给个reward。特别是延迟Delay，常常是几千毫秒之后再返回。

- 深度学习的样本都是独立的，而RL中的state状态却是相关的，前后的状态是有影响的，这显而易见。
- 深度学习的目标分布是固定的。一个图片是车就是车，不会变。但增强学习，分布却是一直变化的，比如超级玛丽，前面的场景和后面的场景不一样，可能前面的训练好了，后面又不行了，或后面的训练好了前面又用不了了。

从上面分析出增强学习要结合深度学习存在的三个问题：

1. 没有标签怎么办?
2. 样本相关性太高怎么办?
3. 目标分布不固定怎么办?

确实，如果没有这篇文章的突破性创新，我们如何知道怎么解决这三个问题。这篇文章至少解决了前两个问题及部分解决了第三个问题。

解决之道 CNN + Q-Learning = Deep Q Network

1. 通过Q-Learning使用reward来构造标签
2. 通过experience replay的方法来解决相关性及非静态分布问题

实验环境

使用Arcade Learning Environment 来训练Atari 2600 游戏。
- 目标：使用一个基于神经网络的agent来学习玩各种游戏，玩的越多越好。
- 输入：要求只输入图像数据和得分，和人类基本一样
- 输出： 控制动作
- 要求：对于不同游戏，网络的结构及顶层参数设定一样

背景知识 Background

要理解这篇文章，没有背景知识是很难的，虽然作者在这里介绍了一下RL的基本知识及Q-learning算法及采用神经网络来代替Q矩阵的方法，但篇幅太短，没有基础很难理解。

核心就是几个公式：Q-learning ，用neural network的loss function，梯度公式。

有了这几个公式支撑，整个算法也就理解一半了。
关于背景知识这一块这里不进行分析了，之后专门进行介绍。

相关工作 Related Work

TD-gammon

看到这里才知道实际上并不是Deepmind第一次将神经网络用于RL，TD-gammon使用了MLP(Multi-layer percetron)也就是一般的神经网络，一个隐藏层（hidden layer）来训练。并且将其应用到了玩backgamm游戏上取得了人类水平。但是很可惜的是，当时人们把算法用到其他游戏象棋围棋并不成功，导致人们为TD-gammon算法只适用于backgammon这个特殊的例子，不具备通用性。

本质上，使用神经网络是为了模拟一个非线性的函数（value或者policy都行，比如flappy bird，设定它上到一个高度下降这就是一个分段函数）。人们发现，将model-free的算法比如Q-learning与非线性函数拟的方法（神经网络是一种）很容易导致Q-network发散。因此，大部分的工作就使用线性的函数拟合（linear function approximation），收敛性好。

其他人

显然不是Deepmind第一个想到把深度学习和增强学习结合在一起的。之前也有人尝试用深度神经网络来计环境environment，估值函数value function或者policy策略。**这实际上是三个Deep Learning与Reinforcement Learning结合的思路**
并且结合Q-learning发散的问题也被**Gradient temporal-difference** 方法部分解决。（这个方法具体是神马有待学习）
这些方法用在使用非线性来估计固定策略或者使用线性来估计一个控制策略还是证明可以收敛的。 但是些方法还没有拓展到非线性控制nonlinear control。

这就是研究点！！！！

最相近的工作 NFQ

采用同样的loss function，但是使用RPROP（不懂）来更新参数，问题是采用batch update而不是sgd 需要多的计算开销而且取决于数据集的大小。

采用deep autoencoder，也是使用visual input。但是不同的是，NFQ是把特征提取和增强学习分开进行的
先提取特征，再应用NFQ训练。
而Deepmind是End-to-End。学习的特征和最后的动作价值是直接关联的。也就是学习什么特征也是网络

关于Atari 2600 模拟器

使用它做增强学习研究之前就有，但采用的是线性函数估计和获取的视觉特征（linear function approximation and generic visual features) 总之之前是人工提取特征，降维。
HyperNEAT使用神经网络来代替一个策略，但不同游戏用不同的网络。

Deep reinforcement learning

目标

当前深度学习的方式核心在于采用大量的数据集，然后使用SGD进行权值的更新。所以，这里的目标就
将增强学习的算法连接到深度神经网络中，然后能直接输入RGB的原始图像，并使用SGD进行处理。

对比TD-gammon的改进之处

实际上TD-gammon的思路就是上面的思路，只是训练是直接获取experience样本进行训练，也就是on-policy。而关键是这个算法是20年前的了。所以，经过20年的硬件发展以及深度学习的发展，没有理由无法在这上面取得突破。

相比于TD-gammon的在线学习方式，Deepmind使用了**experience replay**的技巧。简单的说就是建立一个
验池，把每次的经验都存起来，要训练的时候就 随机 的拿出一个样本来训练。这样就可以解决状态sta
相关的问题。以此同时，动作的选择采用常规的ϵ-greedy policy。 就是小概率选择随机动作，大概率选
最优动作。

然后呢输入的历史数据不可能是随机长度，这里就采用固定长度的历史数据，比如deepmind使用的4帧
像作为一个状态输入。

整个算法就叫做**Deep-Q-Learning**。

Deep-Q-Learning

算法就是如下了：

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    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 $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

算法分析：

1. 训练分成M个episode，每个episode训练T次。我的理解就是比如玩游戏，一局是一个episode，一局里有很多时间片，就训练多少次，次数不固定。重启新的episode主要是初始化state 作为新的第一个，而不用上一局的最后的状态作为state输入。
2. 实际上每个循环分成两部分：一部分是输出动作并存储。一部分是随机从经验池里取出minibatch个transitions，然后计算target，根据loss function通过RMSProp更新参数。（minibatch是什么意思？）
3. 这里的算法我们可以看到，参数是一直更新的，而Nature的算法改进了，计算target用的是之前的参数具体算法的变化等之后分析Nature的文章再说。

算法优点对比standard online Q-learning

- 每一步的经验都能带来很多权值的更新，拥有更高的数据效率（个人不是很理解这作为一个优点以前的算法就没有吗？）
- 就是experience replay的优势，打破数据的相关性，降低数据更新的不确定性variance。
- experience replay的另一个优点就是不容易陷入局部最优解或者更糟糕的不收敛。 如果是on-policy learning，也就是来一个新的经验就学一个。那么下一个动作就会受当前的影响，如果最大的动作向左，那么就会一直向左。使用experience replay 获取的行为的分布就比较平均，就能防止大的波和发散。也因此，这是一个off-policy的学习。

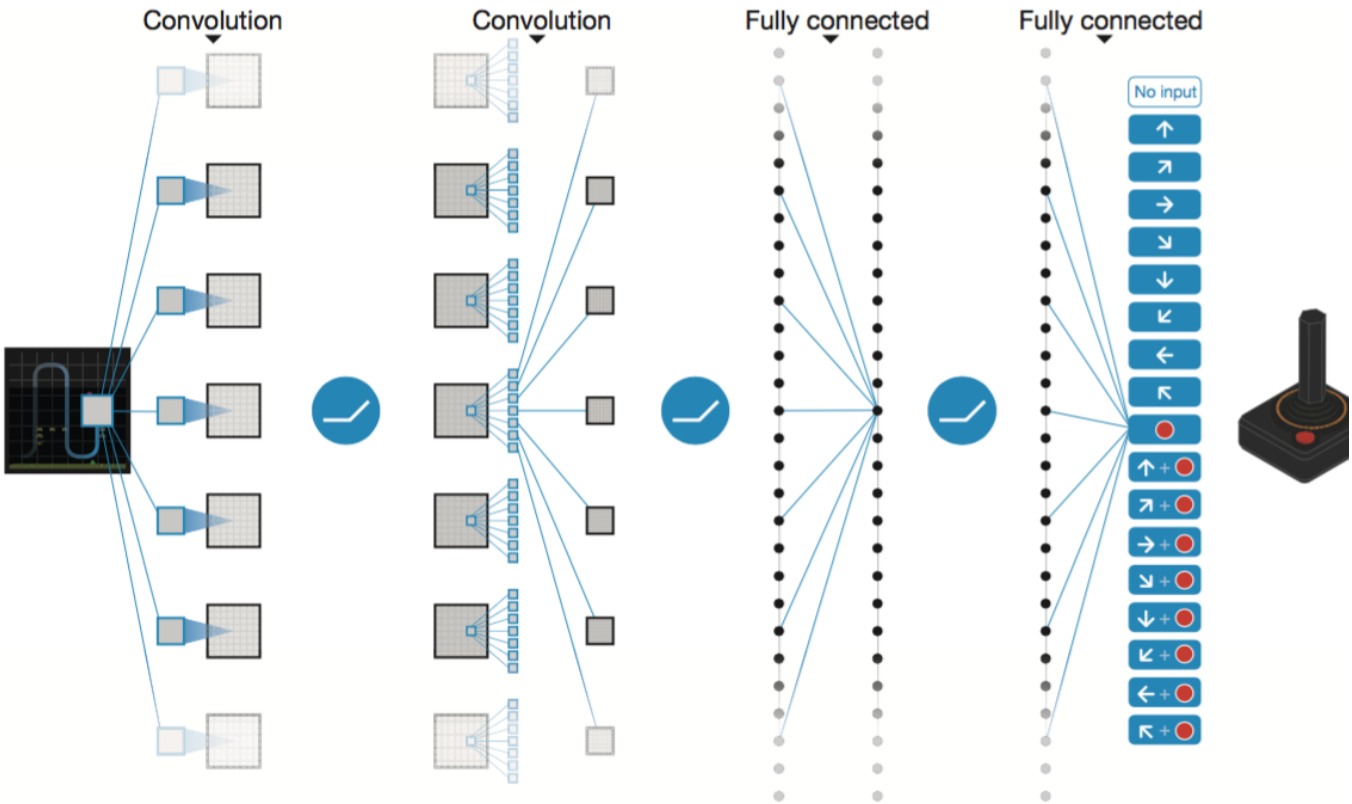
实际应用中，只存储N个经验在经验池里（毕竟空间有限嘛）这个方法的局限性就是这个经验池并没有分重要的转移transition，总是覆盖最新的transition。
所以，采用有优先级的使用memory是一个更好的方式。这也就是阿蒙说的**引导的经验池**。

预处理与网络模型架构

因为输入是RGB，像素也高，因此，对图像进行初步的图像处理，变成灰度矩形84*84的图像作为输入有利于卷积。
接下来就是模型的构建问题，毕竟Q(s,a)包含s和a。一种方法就是输入s和a，输出q值，这样并不方便，个a都需要forward一遍网络。

Deepmind的做法是神经网络只输入s，输出则是每个a对应的q。这种做法的优点就是只要输入s，forwar向传播一遍就可以获取所有a的q值，毕竟a的数量有限。

具体的模型架构如下：



实验

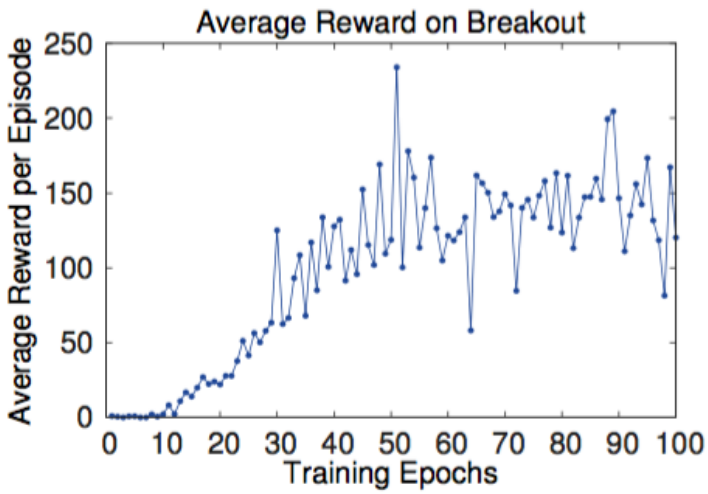
- 测试7个游戏

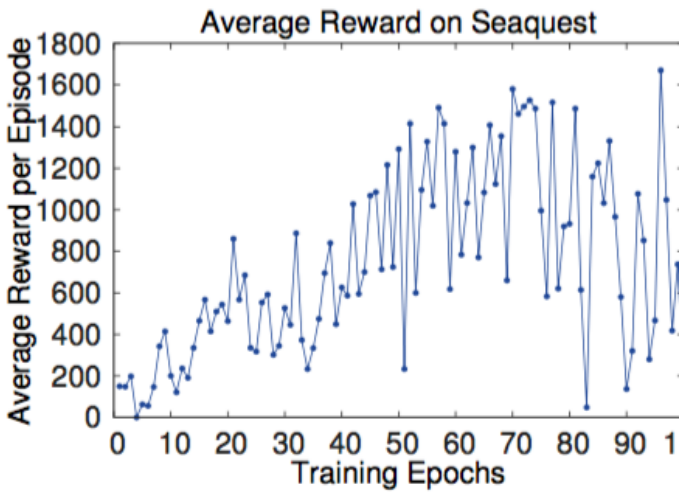
- 统一不同游戏的reward，正的为1，负的为-1，其他为0。这样做a,R的好处是限制误差的比例并且以使用统一的训练速度来训练不同的游戏
- 使用RMSProp算法，就是minibatch gradient descent方法中的一种。Divide the gradient by a running average of its recent magnitude. 梯度下降有很多种方法包括（SGD,Momenturn,NAG,Adagrad,Adadelta,Rmsprop) 相关问题以后再分析。
- ϵ-greedy 前1百万次从1 下降到0.1，然后保持不变。这样一开始的时候就更多的是随机搜索，之后慢使用最优的方法。
- 使用**frame-skipping technique**,意思就是每k frame才执行一次动作，而不是每帧都执行。在实际的究中，如果每帧都输出一个动作，那么频率就太高，基本上会导致失败。在这里，中间跳过的帧用的动作为之前最后的动作。这和人类的行为是一致的，人类的反应时间只有0.1，也是采用同样做法。并且这样做可以提速明显的。那么这里Deepmind大部分是选择k=4，也就是每4帧输出一个作。

训练

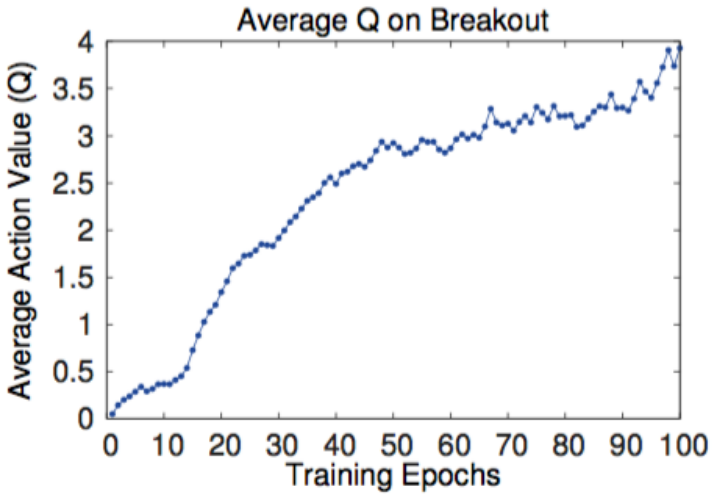
如何在训练的过程中估计训练的效果在RL上是个Challenge。毕竟不像监督学习，可以有training 和validation set。那么只能使用reward，或者说平均的reward来判定。也就是玩的好就是训练的好。

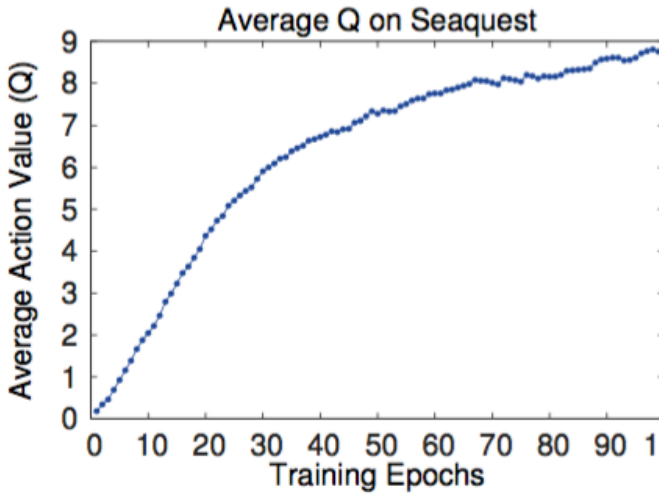
但是存在问题就是reward的噪声很大，因为很小的权值改变都将导致策略输出的巨大变化，从文章的途可以看出：





以此同时，平均Q值的变化却是稳定的，这是必然的，因为每次的Target计算都是使用Q的最大值。：





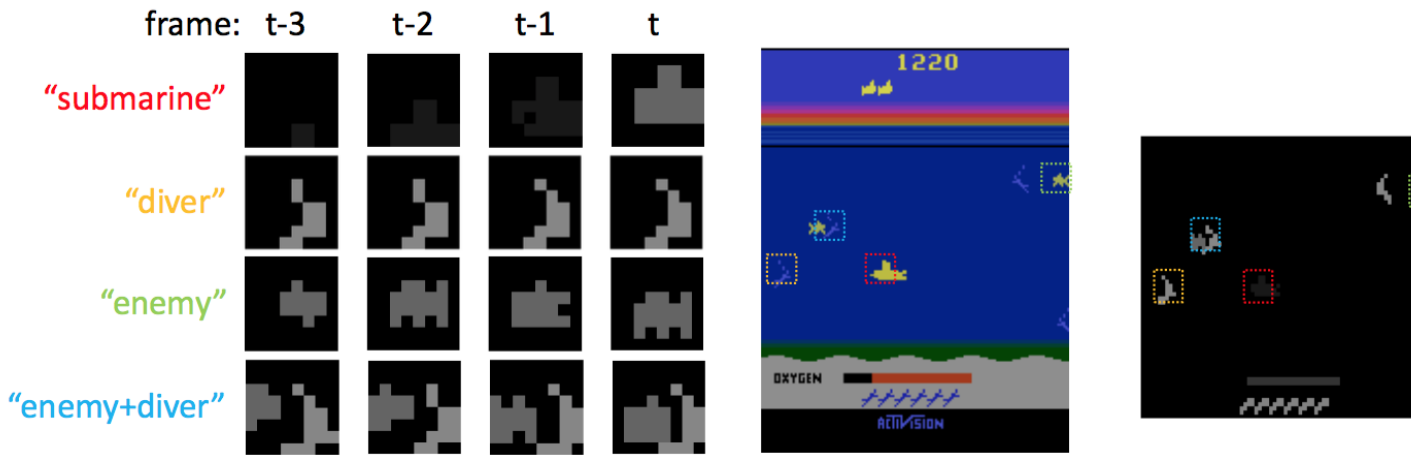
而且很关键的是所有的实验都收敛了！！！

虽然没有理论支持为什么保证收敛，但是就是实现了，Deepmind的方法可以在一个稳定的状态下使用大模的深度神经网络结合增强学习。

显示Value Function

就是看一下每一帧的Q值变化，看了之后答案是惊人的：

在敌人出现时，Q值上升，快消灭敌人时，Q值到顶峰，敌人消失，Q值回到正常水平。这说明Q值确实表了整个复杂的状态。实际上到后面发现，整个神经网络可以同时跟踪多个图上的目标：



cool finding — low level filters show game objects

算法评估

算法对比了。
- Sarsa算法。使用Sarsa算法学习一个线性的policy，采用手工获取的特征。
- Contingency算法。 采用和Sarsa相同的方法，但是通过学习部分屏幕的表达增强了特征。
上面的方法的特征提取都采用传统的图像处理方法比如背景减除。
总之就是特征提取方式落后。Deepmind的算法是原始输入计算机需要自己去detect物体。（直接解决了detection和tracking的问题）

以此同时，当然是对比人类的水平了。人类的得分是人类玩两小时的结果，反正是蛮高的。但deepmin
方法有几个超过人类
对比的列表就不复制了，总而言之就是方法好，原始输入，并且在使用ϵ 为0.05的得分还比其他方法强

总结Conclusion

这篇文章采用了一个全新的方法结合深度学习和增强学习，可以说是deep reinforcement learning的开山
作。采用stochastic minibatch updates以及experience replay的技巧。 效果很强，具有通用性。