用深度强化学习玩雅达利

Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou

Daan Wierstra Martin Riedmiller

DeepMind技术

{弗拉德,koray,大卫,alex.graves, ioannis,她女儿,马丁。Riedmiller} @ deepmind.com

摘要

我们提出了第一个使用强化学习成功地从高维感官输入中直接学习控制策略的深度学习模型。该模型是一个卷积神经网络，用q学习的一种变体进行训练，其输入是原始像素，输出是估计未来奖励的价值函数。我们将我们的方法应用于街机学习环境中的7款雅达利2600游戏，没有调整架构或学习算法。我们发现，它在其中6款游戏上的表现超过了之前所有的方法，并在其中3款游戏上超过了人类专家。

1 介绍

学习直接从高维感官输入(如视觉和语音)中控制代理是强化学习(RL)的长期挑战之一。在这些领域上运行的大多数成功的RL应用程序都依赖于与线性值函数或策略表示相结合的手工制作的特征。显然，这类系统的性能严重依赖于特征表示的质量。

最近深度学习的进展使得从原始感官数据中提取高级特征成为可能，从而导致计算机视觉[11,22,16]和语音识别[6,7]方面的突破。这些方法利用了一系列神经网络架构，包括卷积网络、多层感知机、受限玻尔兹曼机和递归神经网络，并利用了监督学习和无监督学习。人们似乎很自然地会问，类似的技术是否也有利于RL的感官数据。

然而，强化学习从深度学习的角度提出了几个挑战。首先，到目前为止，大多数成功的深度学习应用都需要大量手工标记的训练数据。另一方面，RL算法必须能够从经常稀疏、有噪声和延迟的标量奖励信号中学习。与监督学习中发现的输入和目标之间的直接关联相比，动作和由此产生的奖励之间的延迟，可能长达数千个时间步长，似乎特别令人生畏。另一个问题是，大多数深度学习算法假设数据样本是独立的，而在强化学习中，人们通常会遇到高度相关的状态序列。此外，在RL中，随着算法学习新的行为，数据分布会发生变化，这对于假设固定底层分布的深度学习方法来说可能是有问题的。

本文证明了卷积神经网络可以克服这些挑战，从复杂RL环境中的原始视频数据中学习成功的控制策略。该网络使用q学习[26]算法的一种变体进行训练，使用随机梯度下降来更新权重。为了缓解相关数据和非平稳分布的问题，我们使用

图1:5款雅达利2600游戏的屏幕截图:(从左至右)《Pong》、《Breakout》、《Space Invaders》、《Seaquest》和《Beam Rider》

一个经验回放机制[13]，它随机采样之前的过渡，从而平滑了许多过去行为的训练分布。

我们将这种方法应用于街机学习环境(ALE)中的一系列雅达利2600游戏[3]。雅达利2600是一个具有挑战性的RL测试平台，它为代理提供了高维视觉输入(210 ×160 60Hz RGB视频)和各种有趣的任务，这些任务被设计成对人类玩家来说很困难。我们的目标是创建一个单一的神经网络智能体，能够成功地学习玩尽可能多的游戏。该网络没有被提供任何游戏特定的信息或手工设计的视觉功能，并且不知道模拟器的内部状态;它只从视频输入、奖励和终端信号以及一组可能的动作中学习——就像人类玩家会做的那样。此外，网络结构和用于训练的所有超参数在整个游戏中都保持不变。到目前为止，该网络在我们尝试的七场比赛中有六场超过了之前所有的RL算法，并在其中三场比赛中超过了专业的人类选手。图1提供了其中5款用于训练的游戏的示例截图。

2 背景

考虑智能体与环境E交互的任务，在本例中是Atari模拟器，在一系列动作、观察和奖励中。在每个时间步长，智能体从合法游戏动作集合中选择一个动作a_t，$A = \{1, \cdots K\}$。动作被传递给模拟器，并修改其内部状态和游戏分数。一般E可能是随机的。模拟器的内部状态是不被代理观察到的;相反，它观察来自模拟器的图像$x_t \in R^d$，这是代表当前屏幕的原始像素值的向量。除此之外，它还会收到一个代表游戏分数变化的奖励r_t。注意，一般情况下，游戏得分可能取决于整个先前的行动和观察序列;关于一个动作的反馈可能只有在经过了数千个时间步骤之后才会收到。

由于代理只观察当前屏幕的图像，任务是部分观察的，许多模拟器状态在感知上是别名的，即仅从当前屏幕是不可能完全了解当前情况的x_{t_o} 因此，我们考虑动作和观察的序列，$s_t = x_1$，a_1，x_2，$\cdots$，a_{t-1}，x_t，并学习依赖于这些序列的游戏策略。模拟器中的所有序列都假定在有限的时间步内终止。这种形式主义产生了一个大型但有限的马尔可夫决策过程(MDP)，其中每个序列都是一个不同的状态。因此，我们可以对mdp应用标准的强化学习方法，只需使用完整序列s_t作为时间t的状态表示。

智能体的目标是通过以最大化未来奖励的方式选择动作，与模拟器进行交互。我们做出标准假设，未来奖励按每时间步长的γ因子折现，并将时间t的未来折现*回报*定义为$R_t = \sum_{t0=t}^{T} \gamma^{t0-t} r_{t0}$，其中T为游戏结束的时间步长。我们将最优动作价值函数$Q^*(s, a)$定义为在看到一些序列s之后，然后采取一些行动a，$Q^*(s, a) = \max_\pi E[R_t | s_t = s, a_t = a, \pi]$，其中π是一个将序列映射到行动(或行动上的分布)的策略。

最优动作值函数遵循一个重要的恒等式，即*贝尔曼方程*。这是基于以下直觉:如果序列s^0在下一个时间步的最优值$Q^*(s^0, a^0)$对于所有可能的动作a^0都是已知的，那么最优策略是选择动作a^0

最大化r + γQ的期望值*(s0， a0),

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s',a') \middle| s, a \right] \quad (1)$$

许多强化学习算法背后的基本思想是通过使用贝尔曼方程作为迭代更新来估计动作值函数，$Q_{i+1}(s, a) = E [r + \gamma \max_{a_0} Q_i(s_0, a0)|s, a]$。这样的*值迭代*算法收敛到最优的动作价值函数，$Q_i \rightarrow Q^*$asi→∞[23]。在实践中，这种基本的方法是完全不切实际的，因为动作价值函数是针对每个序列单独估计的，没有任何泛化。相反，常用的方法是使用函数逼近器来估计动作价值函数，$Q(s, a;\theta) \approx Q^*(s, a)$。在强化学习社区中，这通常是一个线性函数逼近器，但有时会使用一个非线性函数逼近器来代替，例如神经网络。我们把权重为θ的神经网络函数逼近器称为q网络。q网络可以通过最小化损失函数序列$L_i(\theta_i)$来训练，损失函数在每次迭代i时都会变化，

$$L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot)} \left[(y_i - Q(s,a;\theta_i))^2 \right], \quad (2)$$

其中$yi = E_{s_0 \boxtimes E}[r + \gamma \max a_0 Q(s0, a0;\theta i-1)|s, a]$是迭代I的目标，ρ(s, a)是序列s和动作a上的概率分布，我们称之为*行为分布*。在优化损失函数$L_i(\theta_i)$.)时，来自上一次迭代θ_{i-1}的参数保持固定不变。注意，目标取决于网络权重;这与监督学习中使用的目标形成对比，在学习开始之前，目标是固定的。根据我们得到的以下梯度的权重对损失函数进行微分，

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a \sim \rho(\cdot); s' \sim \mathcal{E}} \left[\left(r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i) \right) \nabla_{\theta_i} Q(s,a;\theta_i) \right]. \quad (3)$$

而不是在上面的梯度中计算全部期望，通过随机梯度下降来优化损失函数通常在计算上是方便的。如果在每个时间步后更新权重，并且期望分别被来自行为分布ρ和模拟器E的单个样本替换，那么我们就得到了熟悉的*Q-learning*算法[26]。

注意，这个算法是*无模型的:*它直接使用来自模拟器E的样本来解决强化学习任务，而没有显式构造E的估计。它也是*off-policy:*它学习贪婪策略$a = \max_a Q(s, a;\theta)$，同时遵循确保对状态空间进行充分探索的行为分布。在实践中，行为分布通常是由ε-greedy策略选择的，该策略遵循概率为1−ε的贪婪策略，并选择概率为ε的随机行动。

3 相关工作

也许强化学习最著名的成功案例是*TD-gammon*，这是一个完全通过强化学习和自我游戏学习的双陆棋游戏程序，并达到了超人的游戏水平[24]。TD-gammon使用了类似于q学习的无模型强化学习算法，并使用带有一个隐藏层的多层感知器近似值函数1。

然而，早期对TD-gammon的后续尝试，包括将同样的方法应用于国际象棋、围棋和跳棋，都不太成功。这导致人们普遍认为，TD-gammon方法是一种特殊情况，只适用于双陆棋，可能是因为掷骰子的随机性有助于探索状态空间，也使价值函数特别平滑[19]。

此外，研究表明，将无模型强化学习算法(如q学习)与非线性函数逼近器[25]或离策略学习[1]相结合，可能会导致q网络发散。随后，强化学习的大部分工作都集中在具有更好收敛性的线性函数逼近器[25]上。

[1] In fact TD-Gammon approximated the state value function V (s) rather than the action-value function Q(s, a), and learnt *on-policy* directly from the self-play games

最近，人们又重新燃起了将深度学习与强化学习相结合的兴趣。深度神经网络已经被用来估计环境E;受限玻尔兹曼机已被用于估计价值函数[21];或政策[9]。此外，*梯度时间差分*方法已经部分解决了q学习的散度问题。当用非线性函数逼近器[14]评估固定策略时，这些方法被证明是收敛的;或者使用q学习的受限变体学习线性函数逼近的控制策略[15]。然而，这些方法还没有扩展到非线性控制。

也许与我们自己的方法最相似的先前工作是神经拟合q学习(NFQ)[20]。NFQ对式2中的损失函数序列进行优化，使用RPROP算法更新q网络的参数。然而，它使用的是批量更新，每次迭代的计算成本与数据集的大小成正比，而我们考虑的是随机梯度更新，每次迭代的常量成本较低，并可扩展到大型数据集。NFQ也已成功应用于使用纯视觉输入的简单现实世界控制任务，首先使用深度自编码器学习任务的低维表示，然后将NFQ应用于该表示[12]。相比之下，我们的方法应用了端到端的强化学习，直接从视觉输入;因此，它可能会学习到与判别动作价值直接相关的特征。q学习以前也与经验回放和简单的神经网络相结合[13]，但再次从低维状态开始，而不是原始的视觉输入。

雅达利 2600模拟器作为强化学习平台的使用由[3]提出，该模拟器采用了具有线性函数近似和通用视觉特征的标准强化学习算法。随后，通过使用更大数量的特征，并使用拔河哈希将特征随机投影到低维空间[2]来改进结果。HyperNEAT进化架构[8]也被应用于雅达利平台，在那里它被用来进化(单独地，为每个不同的游戏)一个代表该游戏策略的神经网络。当使用模拟器的重置功能对确定性序列进行反复训练时，这些策略能够利用几个雅达利游戏中的设计缺陷。

4 深度强化学习

最近在计算机视觉和语音识别方面的突破依赖于在非常大的训练集上有效地训练深度神经网络。最成功的方法是直接从原始输入中进行训练，使用基于随机梯度下降的轻量级更新。通过向深度神经网络输入足够的数据，通常可以学习到比手工特征[11]更好的表示。这些成功激发了我们对强化学习的方法。我们的目标是将强化学习算法连接到一个深度神经网络，该网络直接对RGB图像进行操作，并通过使用随机梯度更新有效地处理训练数据。

Tesauro的TD-Gammon架构为这种方法提供了一个起点。这个架构更新了一个估计价值函数的网络的参数，直接来自于经验的on-policy样本s_t，a_t，r_t，s_{t+1}，a_{t+1}，从算法与环境的交互中提取(或通过自玩，在backgammon的情况下)。由于这种方法能够在20年前超越人类最好的双陆棋玩家，人们自然会想，20年的硬件改进，再加上现代的深度神经网络架构和可扩展的RL算法，是否可能产生显著的进步。

与TD-Gammon和类似的在线方法相比，我们利用了一种称为经验*回放*的技术[13]，我们将代理在每个时间步的经验，$e_t=(s_t, a_t, r_t, s_{t+1})$存储在数据集$D = e_1, \cdots, e_N$，将多个片段汇集到一个*回放记忆中*。在算法的内循环期间，我们将q学习更新或小批量更新应用于从存储样本池中随机抽取的经验样本$e \sim D$。在执行经验回放后，智能体根据-greedy策略选择并执行一个动作。由于使用任意长度的历史作为神经网络的输入可能很困难，因此我们的q函数转而处理由函数ϕ产生的固定长度的历史表示。完整的算法，我们称之为*深度q学习*，在算法1中给出。

与标准的在线q学习相比，这种方法有几个优点[23]。首先，每一步的经验都可能用于许多重量更新，这允许更大的数据效率。

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

第二，直接从连续的样本中学习效率低下，这是因为样本之间的强相关性;随机化样本破坏了这些相关性，因此降低了更新的方差。第三，在策略学习时，当前参数决定了参数训练的下一个数据样本。例如，如果最大化的行动是向左移动，那么训练样本将由左手边的样本支配;如果最大化动作然后切换到右边，那么训练分布也会切换。很容易看到不必要的反馈循环可能会出现，参数可能会陷入一个糟糕的局部最小值，甚至灾难性地发散到[25]。通过使用经验回放，行为分布在其先前的许多状态上进行平均，平滑学习并避免参数中的振荡或发散。请注意，当通过经验重放学习时，有必要进行非策略学习(因为我们当前的参数与用于生成样本的参数不同)，这激发了q学习的选择。

在实践中，我们的算法只将最后N个经验元组存储在回放内存中，并在执行更新时从D中均匀随机采样。这种方法在某些方面是有限的，因为内存缓冲区不区分重要的转换，并且由于内存大小n有限，总是用最近的转换重写。类似地，均匀采样对重播内存中的所有转换赋予同等的重要性。一个更复杂的采样策略可能会强调我们可以从中学习到最多的转换，类似于优先扫描[17]。

4.1 预处理和模型架构

直接使用原始的Atari帧，即210 ×160像素图像和128个调色板，可能需要计算，因此我们应用了一个基本的预处理步骤，旨在降低输入维度。通过首先将原始帧的RGB表示转换为灰度，并将其下采样为110×84图像，对原始帧进行预处理。最终的输入表示是通过裁剪图像的84 × 84区域来获得的，该区域大致捕捉了比赛区域。最后的裁剪阶段是必须的，因为我们使用了来自[11]的2D卷积的GPU实现，它期望正方形输入。对于本文的实验，算法1中的函数ϕ将此预处理应用于历史的最后4帧，并将它们堆叠以产生q函数的输入。

有几种可能的方法可以使用神经网络对Q进行参数化。由于Q将历史-动作对映射为其q值的标量估计，因此以前的一些方法已将历史和动作用作神经网络的输入[20,12]。这种类型架构的主要缺点是需要单独的前向传递来计算每个动作的q值，从而导致成本与动作数量呈线性增长。相反，我们使用了一种架构，其中每个可能的动作都有一个单独的输出单元，只有状态表示是神经网络的输入。输出对应于输入状态的单个动作的预测q值。这种类型的体系结构的主要优点是能够计算给定状态下所有可能的操作的q值，只需通过网络进行一次转发。

现在我们将描述所有七款雅达利游戏所使用的确切架构。神经网络的输入由φ制作的84 ×84 ×4图像组成。第一个隐藏层将16个带步幅4的8 ×8滤波器与输入图像卷积，并应用整流器非线性[10,18]。第二个隐藏层用步幅2卷积32个4 ×4滤波器，接下来又是一个整流器非线性。最后一个隐藏层是全连接的，由256个整流器单元组成。输出层是一个全连接的线性层，每个有效动作都有一个输出。在我们考虑的游戏中，有效动作的数量在4到18之间变化。我们将使用我们的方法训练的卷积网络称为深度q网络(DQN)。

5 实验

到目前为止，我们已经在7款受欢迎的ATARI游戏上进行了实验——beam Rider, Breakout, Enduro, Pong, Q*bert, Seaquest，太空入侵者。我们在所有7款游戏中使用了相同的网络架构、学习算法和超参数设置，表明我们的方法足够稳健，可以在不纳入游戏特定信息的情况下适用于各种游戏。虽然我们在真实的和未修改的游戏上评估了我们的代理，但我们仅在训练期间对游戏的奖励结构做了一个更改。由于不同游戏的分数范围差异很大，我们将所有正奖励固定为1，所有负奖励固定为−1，保持0奖励不变。以这种方式裁剪奖励限制了误差导数的规模，并使在多个游戏中使用相同的学习率更容易。同时，它可能会影响我们的智能体的性能，因为它无法区分不同量级的奖励。
在这些实验中，我们使用了大小为32的minibatches的RMSProp算法。训练期间的行为策略是-greedy，在前100万帧中从1线性退火到0.1，此后固定在0.1。我们总共训练了1000万帧，并使用了100万最近帧的回放记忆。

根据之前玩雅达利游戏的方法，我们也使用了一种简单的跳帧技术[3]。更准确地说，代理在每个k^{th}帧上看到并选择动作，而不是每一帧，并且它的最后一个动作在跳过的帧上重复。由于向前运行模拟器一步所需的计算比让代理选择一个动作少得多，这种技术允许代理玩大约k倍的游戏，而不会显著增加运行时间。我们在所有游戏中都使用k = 4，但《太空入侵者》除外，我们注意到使用k = 4会使激光因闪烁周期而不可见。我们使用k = 3使激光可见，这一变化是任何游戏之间的超参数值的唯一区别。

5.1 训练和稳定性

在监督学习中，通过在训练集和验证集上评估模型，可以很容易地跟踪模型在训练期间的性能。然而，在强化学习中，准确评估一个智能体在训练期间的进展可能具有挑战性。由于我们的评估指标，如[3]所建议的，是智能体在一个episode或游戏中收集的在多个游戏中的平均总奖励，我们在训练期间定期计算它。平均总奖励指标往往非常嘈杂，因为策略权重的小变化可能导致策略访问的状态分布的大变化。图2中最左边的两个图显示了在Seaquest和Breakout两款游戏的训练过程中，平均总奖励是如何演变的。这两个平均奖励图确实相当嘈杂，给人一种学习算法没有取得稳定进展的印象。另一个更稳定的指标是策略的估计动作价值函数Q，它提供了一个估计，即在任何给定状态下，智能体通过遵循其策略可以获得多少折扣奖励。我们通过在训练开始之前运行一个随机策略来收集一组固定的状态，并跟踪这些状态的最大[2]预测Q的平均值。图2中最右边的两幅图显示，平均预测Q的增长比智能体获得的平均总奖励更平滑，在其他五款游戏上绘制相同的指标会产生类似的平滑曲线。除了在训练期间看到预测Q的相对平滑的提高外，我们在任何实验中都没有遇到任何分歧问题。这表明，尽管缺乏任何理论收敛保证，我们的方法能够以稳定的方式使用强化学习信号和随机梯度下降来训练大型神经网络。

[2] The maximum for each state is taken over the possible actions.

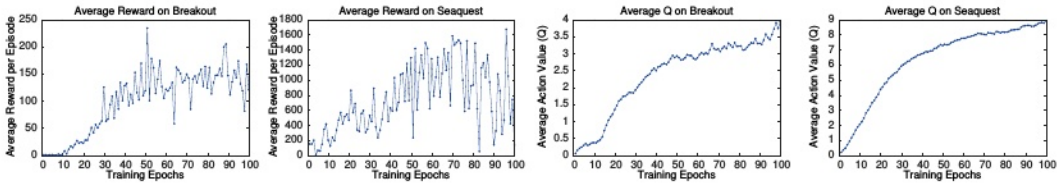

图2:左边的两个图分别显示了训练期间《越狱》和《Seaquest》每集的平均奖励。统计数据是通过运行-greedy策略，= 0.05进行10000步计算得到的。右边的两个图分别显示了在突破和Seaquest状态下的一组状态的平均最大预测动作值。一个epoch对应50000个小批量重量更新或大约30分钟的训练时间。

图3:最左边的图显示了游戏Seaquest的30帧片段的预测值函数。这三张截图分别对应标有A、B、C的帧。

5.2 可视化价值函数

图3显示了游戏《Seaquest》的学习值函数的可视化。图表明,预测值跳跃后敌人出现在屏幕的左边(A)。代理然后在敌人发射鱼雷和预报值的峰值随着鱼雷击中敌人(B点)。最后,大约值落在其原始值后敌人消失(C点)。图3表明,我们的方法可以学习的价值函数的发展相当复杂的事件序列。

5.3 主要评价

我们将我们的结果与RL文献[3,4]中表现最好的方法进行了比较。标记为Sarsa的方法使用Sarsa算法来学习针对雅达利任务手工设计的几个不同特征集的线性策略，我们报告了表现最佳的特征集的分数[3]。偶然性使用了与Sarsa相同的基本方法，但通过对智能体控制下的屏幕部分的学习表示来增强特征集[4]。请注意，这两种方法都通过使用背景减法和将128种颜色中的每一种视为一个单独的通道来纳入关于视觉问题的重要先验知识。由于许多雅达利游戏对每种类型的对象使用一种不同的颜色，因此将每种颜色视为单独的通道，就类似于生成编码每种对象类型的单独二进制地图。相比之下，我们的智能体只接收原始的RGB截图作为输入，并且必须学会自己检测物体。

除了学习的智能体，我们还报告了一个专家人类游戏玩家的分数和一个随机均匀选择动作的策略。人类的表现是在玩每个游戏约两个小时后获得的中位数奖励。请注意，我们报告的人类得分远高于Bellemare等人[3]的得分。对于学习到的方法，我们遵循Bellemare等人[3,5]中使用的评估策略，报告通过运行= 0.05的-greedy策略在固定步数下获得的平均分数。表1的前五行显示了所有游戏的每场平均得分。我们的方法(标记为DQN)在所有七场比赛中都以显著优势优于其他学习方法，尽管几乎没有纳入关于输入的先验知识。

我们还在表1的最后三行中对[8]的进化策略搜索方法进行了比较。我们报告了这种方法的两组结果。HNeat最佳分数反映了使用手工设计的对象检测器算法获得的结果，该算法输出位置和

	B. Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S. Invaders
Random	354	1.2	0	−20.4	157	110	179
Sarsa [3]	996	5.2	129	−19	614	665	271
Contingency [4]	1743	6	159	−17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	−3	18900	28010	3690
HNeat Best [8]	3616	52	106	19	1800	920	**1720**
HNeat Pixel [8]	1332	4	91	−16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	1075

表1:上面的表比较了在固定步数下，通过运行= 0.05的-greedy策略，各种学习方法的平均总奖励。下表报告了HNeat和DQN单集最佳表现的结果。HNeat产生的确定性策略总是得到相同的分数，而DQN使用= 0.05的-greedy策略。

雅达利屏幕上的物体类型。HNeat像素分数是通过使用雅达利模拟器的特殊8色通道表示获得的，该模拟器表示每个通道上的对象标签映射。这种方法在很大程度上依赖于找到一个表示成功漏洞攻击的确定的状态序列。以这种方式学习到的策略不太可能泛化到随机扰动;因此，该算法只在得分最高的单集上进行评估。相比之下，我们的算法是在-greedy控制序列上进行评估的，因此必须在各种可能的情况下泛化。尽管如此，我们发现除了《太空入侵者》，在所有游戏中，不仅我们的最大评估结果(第8行)，而且我们的平均结果(第4行)都获得了更好的表现。

最后，我们表明，我们的方法在Breakout、Enduro和Pong上取得了比人类专家玩家更好的性能，并且在Beam Rider上取得了接近人类的性能。《Q*bert》、《Seaquest》和《太空入侵者》等游戏的表现与人类相差很远，它们更具挑战性，因为它们要求神经网络找到一种长期延伸的策略。

6 结论

本文介绍了一种新的用于强化学习的深度学习模型，并展示了它在仅使用原始像素作为输入的情况下掌握雅达利2600电脑游戏的困难控制策略的能力。我们还提出了在线Q-learning的一个变体，它将随机minibatch更新与经验回放记忆相结合，以简化RL的深度网络训练。我们的方法在测试的7款游戏中的6款中给出了最先进的结果，没有调整架构或超参数。

参考文献

[1] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Machine Learning (ICML 1995)*, pages 30–37. Morgan Kaufmann, 1995.

[2] Marc Bellemare, Joel Veness, and Michael Bowling. Sketch-based linear value function approximation. In *Advances in Neural Information Processing Systems 25*, pages 2222–2230, 2012.

[3] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[4] Marc G Bellemare, Joel Veness, and Michael Bowling. Investigating contingency awareness using atari 2600 games. In *AAAI*, 2012.

[5] Marc G. Bellemare, Joel Veness, and Michael Bowling. Bayesian learning of recursively factored environments. In *Proceedings of the Thirtieth International Conference on Machine Learning (ICML 2013)*, pages 1211–1219, 2013.

[6] George E. Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30 –42, January 2012.

[7] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. In *Proc. ICASSP*, 2013.

[8] Matthew Hausknecht, Risto Miikkulainen, and Peter Stone. A neuro-evolution approach to general atari game playing. 2013.

[9] Nicolas Heess, David Silver, and Yee Whye Teh. Actor-critic reinforcement learning with energy-based policies. In *European Workshop on Reinforcement Learning*, page 43, 2012.

[10] Kevin Jarrett, Koray Kavukcuoglu, MarcAurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, pages 2146–2153. IEEE, 2009.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.

[12] Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.

[13] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, DTIC Document, 1993.

[14] Hamid Maei, Csaba Szepesvari, Shalabh Bhatnagar, Doina Precup, David Silver, and Rich Sutton. Convergent Temporal-Difference Learning with Arbitrary Smooth Function Approximation. In *Advances in Neural Information Processing Systems 22*, pages 1204–1212, 2009.

[15] Hamid Maei, Csaba Szepesv´ari, Shalabh Bhatnagar, and Richard S. Sutton. Toward off-policy learning control with function approximation. In *Proceedings of the 27th International Con-ference on Machine Learning (ICML 2010)*, pages 719–726, 2010.

[16] Volodymyr Mnih. *Machine Learning for Aerial Image Labeling*. PhD thesis, University of Toronto, 2013.

[17] Andrew Moore and Chris Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, 1993.

[18] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, pages 807–814, 2010.

[19] Jordan B. Pollack and Alan D. Blair. Why did td-gammon work. In *Advances in Neural Information Processing Systems 9*, pages 10–16, 1996.

[20] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *Machine Learning: ECML 2005*, pages 317–328. Springer, 2005.

[21] Brian Sallans and Geoffrey E. Hinton. Reinforcement learning with factored states and actions. *Journal of Machine Learning Research*, 5:1063–1088, 2004.

[22] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR 2013)*. IEEE, 2013.

[23] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[24] Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[25] John N Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. *Automatic Control, IEEE Transactions on*, 42(5):674–690, 1997.

[26] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.