

# 使用深度强化学习玩Atari游戏

弗拉基米尔·米尼 科拉伊·卡武克库奥卢 大卫·西尔弗 亚历克斯·格雷夫斯 约安尼斯·安东诺格鲁

达安·维尔斯特拉 马丁·里德米勒

深度思维科技

{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

## 摘要

我们提出了首个成功利用强化学习直接从高维感官输入中学习控制策略的深度强化学习模型。该模型是一个卷积神经网络，采用Q-learning的变体进行训练，其输入为原始像素，输出为估计未来奖励的价值函数。我们将该方法应用于来自Arcade Learning Environment的七款Atari 2600游戏，未对架构或学习算法进行调整。我们发现，在六款游戏中，它的表现优于所有先前的方法，并在其中三款游戏中超越了人类专家。

## 1 引言

学习直接从视觉和语音等高维感官输入控制智能体是强化学习（RL）长期面临的挑战之一。在这些领域运行的大多数成功RL应用都依赖于手工设计的特征与线性价值函数或策略表示的结合。显然，此类系统的性能在很大程度上取决于特征表示的质量。

深度学习的最新进展使得从原始感官数据中提取高级特征成为可能，从而在计算机视觉[11, 22, 16]和语音识别[6, 7]领域取得了突破。这些方法利用了多种神经网络架构，包括卷积网络、多层感知器、受限玻尔兹曼机和循环神经网络，并同时利用了监督学习和无监督学习。很自然地，我们会问类似的技术是否也能对使用感官数据的强化学习（RL）带来益处。

然而，从深度学习的角度来看，强化学习提出了若干挑战。首先，迄今为止大多数成功的深度学习应用都需要大量手工标注的训练数据。而强化学习算法则必须能够从通常是稀疏、噪声大且延迟的标量奖励信号中学习。动作与随之产生的奖励之间的延迟可能长达数千个时间步长，与监督学习中输入与目标之间的直接关联相比，这显得尤为艰巨。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，通常会遇到高度相关的状态序列。此外，在强化学习中，数据分布会随着算法学习新行为而改变，这对于假设基础分布固定的深度学习方法来说可能是个问题。

本文证明，卷积神经网络能够克服这些挑战，在复杂的强化学习环境中从原始视频数据中学习成功的控制策略。该网络通过Q-learning [26]算法的一个变体进行训练，并使用随机梯度下降来更新权重。为了缓解数据相关性和非平稳分布的问题，我们采用



图1：五款Atari 2600游戏的截图：(Left-to-right) 乒乓球、打砖块、太空侵略者、海底大战、光束骑士

一种经验回放机制 [13]，它随机采样先前的转换，从而平滑了训练分布，涵盖了许多过去的行为。

我们将我们的方法应用于一系列在街机学习环境 (ALE) [3]中实现的Atari 2600游戏。Atari 2600是一个具有挑战性的强化学习测试平台，它为智能体提供了高维视觉输入 ( $210 \times 160$  RGB视频, 60Hz) 以及一系列多样且有趣的任务，这些任务设计初衷是让人类玩家感到困难。我们的目标是创建一个单一的神经网络智能体，能够成功学习并尽可能多地玩这些游戏。该网络没有提供任何游戏特定的信息或手工设计的视觉特征，也没有访问模拟器的内部状态；它仅从视频输入、奖励和终止信号以及可能的动作集中学习——就像人类玩家一样。此外，用于训练的网络架构和所有超参数在游戏中保持不变。到目前为止，该网络在我们尝试的七款游戏中的六款上表现优于所有先前的强化学习算法，并在其中三款游戏中超越了人类专家玩家。图1提供了用于训练的五款游戏的示例截图。

## 2 背景

我们考虑的任务中，代理与环境 $\mathcal{E}$  (在此情况下为Atari模拟器) 通过一系列动作、观察和奖励进行交互。在每个时间步，代理从合法游戏动作集合 $\mathcal{A} = \{1, \dots, K\}$ 中选择一个动作 $a_t$ 。该动作传递给模拟器，并修改其内部状态和游戏得分。通常 $\mathcal{E}$ 可能是随机的。模拟器的内部状态不被代理观察；相反，它观察到来自模拟器的一个图像 $x_t \in \mathbb{R}^d$ ，这是代表当前屏幕的原始像素值向量。此外，它接收一个奖励 $r_t$ ，代表游戏得分的变化。需要注意的是，通常游戏得分可能依赖于整个先前的动作和观察序列；关于一个动作的反馈可能只有在经过数千个时间步之后才能收到。

由于智能体仅能观察到当前屏幕的图像，任务处于部分可观测状态，且许多模拟器状态在感知上存在混淆，即仅凭当前屏幕 $x_t$ 无法完全理解当前情况。因此，我们考虑动作和观测的序列 $s_t = x_1, a_1, x_2, \dots, a_{t-1}, x_t$ ，并学习依赖于这些序列的游戏策略。假设模拟器中的所有序列都在有限的时间步长内终止。这种形式化方法产生了一个庞大但有限的马尔可夫决策过程 (MDP)，其中每个序列都是一个独立的状态。因此，我们可以通过简单地使用完整序列 $s_t$ 作为时间 $t$ 的状态表示，来应用标准的MDP强化学习方法。

智能体的目标是通过选择动作的方式与模拟器交互，以最大化未来奖励。我们做出标准假设，即未来奖励每时间步按因子 $\gamma$ 进行折现，并将时间 $t$ 处的未来折现return定义为 $R_t = \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$ ，其中 $T$ 是游戏终止的时间步。我们将最优动作值函数 $Q^*(s, a)$ 定义为在观察到某个序列 $s$ 后采取某个动作 $a$ ，通过遵循任何策略可获得的最大期望回报，即 $Q^*(s, a) = \max_{\pi} \mathbb{E}[R_t | s_t = s, a_t = a, \pi]$ ，其中 $\pi$ 是将序列映射到动作 (或动作分布) 的策略。

最优动作值函数遵循一个重要的恒等式，称为Bellman equation。这是基于以下直觉：如果已知下一个时间步所有可能动作 $a'$ 的序列 $s'$ 的最优值 $Q^*(s', a')$ ，那么最优策略就是选择动作 $a'$ 。

最大化  $r + \gamma Q^*(s', a')$  的期望值，

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

许多强化学习算法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， $Q_{i+1}(s, a) = \mathbb{E} [r + \gamma \max_{a'} Q_i(s', a') \mid s, a]$ 。这样的 *value iteration* 算法会收敛到最优动作值函数， $Q_i \rightarrow Q^*$  如  $i \rightarrow \infty$  [23]。实际上，这种基本方法完全不切实际，因为动作值函数是为每个序列单独估计的，没有任何泛化能力。相反，通常使用函数逼近器来估计动作值函数， $Q(s, a; \theta) \approx Q^*(s, a)$ 。在强化学习社区中，这通常是一个线性函数逼近器，但有时也会使用非线性函数逼近器，例如神经网络。我们将具有权重  $\theta$  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一系列在每次迭代  $i$  时变化的损失函数  $L_i(\theta_i)$  来进行训练。

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

其中  $y_i = \mathbb{E}_{s' \sim \mathcal{E}} [r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) \mid s, a]$  是迭代  $i$  的目标， $\rho(s, a)$  是序列  $s$  和动作  $a$  上的概率分布，我们称之为 *behaviour distribution*。在优化损失函数  $L_i(\theta_i)$  时，前一次迭代  $\theta_{i-1}$  的参数保持不变。请注意，目标依赖于网络权重；这与用于监督学习的目标不同，后者在学习开始前就已固定。对损失函数关于权重求导，我们得到以下梯度，

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

与其计算上述梯度中的完整期望，通常更计算上便捷的是通过随机梯度下降来优化损失函数。如果在每个时间步之后更新权重，并将期望分别替换为行为分布  $\rho$  和模拟器  $\mathcal{E}$  的单个样本，那么我们会得到熟悉的 *Q-learning* 算法[26]。

请注意，该算法是 *model-free*：它直接使用来自模拟器  $\mathcal{E}$  的样本来解决强化学习任务，而无需显式构建  $\mathcal{E}$  的估计。它也是 *off-policy*：在学习贪婪策略  $a = \max_a Q(s, a; \theta)$  的同时，遵循一种确保充分探索状态空间的行为分布。在实践中，行为分布通常通过  $\epsilon$ -贪婪策略选择，该策略以概率  $1 - \epsilon$  遵循贪婪策略，并以概率  $\epsilon$  选择随机动作。

### 3 相关工作

或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对弈学习的双陆棋程序，并达到了超越人类的水平[24]。TD-gammon 使用了类似于 Q 学习的无模型强化学习算法，并通过具有一个隐藏层的多层感知器来近似价值函数<sup>1</sup>。

然而，早期对 TD-gammon 的后续尝试，包括将相同方法应用于国际象棋、围棋和跳棋，都不太成功。这导致了一种普遍观点，即 TD-gammon 方法是一个特例，仅在双陆棋中有效，或许是因为骰子掷出的随机性有助于探索状态空间，同时也使得价值函数特别平滑[19]。

此外，研究表明，将无模型强化学习算法（如 Q-learning）与非线性函数近似器[25]结合，或者实际上与离策略学习[1]结合，可能导致 Q 网络发散。随后，强化学习领域的大部分工作都集中在具有更好收敛保证的线性函数近似器上[25]。

<sup>1</sup>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，将深度学习与强化学习结合的兴趣再度兴起。深度神经网络已被用于估计环境  $\mathcal{E}$ ；受限玻尔兹曼机被用于估计价值函数 [21] 或策略 [9]。此外，Q 学习的发散问题已通过 *gradient temporal-difference* 方法得到部分解决。这些方法在评估固定策略时，使用非线性函数逼近器已被证明能够收敛 [14]；或者在使用线性函数逼近学习控制策略时，通过 Q 学习的受限变体也能收敛 [15]。然而，这些方法尚未扩展到非线性控制领域。

或许与我们方法最为相似的先前工作是神经拟合 Q 学习 (NFQ) [20]。NFQ 通过优化公式 2 中的损失函数序列，利用 RPROP 算法更新 Q 网络的参数。然而，它采用批量更新方式，每次迭代的计算成本与数据集大小成正比，而我们考虑的是每次迭代具有较低恒定成本并能扩展至大规模数据集的随机梯度更新。NFQ 还成功应用于仅依赖视觉输入的简单现实世界控制任务，首先通过深度自编码器学习任务的低维表示，然后对该表示应用 NFQ [12]。相比之下，我们的方法直接从视觉输入端到端地应用强化学习；因此，它可能学习到与区分动作值直接相关的特征。Q 学习也曾与经验回放和简单神经网络结合使用 [13]，但同样是从低维状态而非原始视觉输入开始。

使用 Atari 2600 模拟器作为强化学习平台是由 [3] 引入的，他们应用了带有线性函数逼近和通用视觉特征的标准强化学习算法。随后，通过使用更多特征以及利用拔河哈希将特征随机投影到低维空间，结果得到了改善 [2]。HyperNEAT 进化架构 [8] 也被应用于 Atari 平台，用于为每个不同的游戏分别进化出一个代表该游戏策略的神经网络。当通过模拟器的重置功能反复针对确定性序列进行训练时，这些策略能够利用多个 Atari 游戏中的设计缺陷。

## 4 深度强化学习

计算机视觉和语音识别领域的最新突破依赖于在非常大的训练集上高效训练深度神经网络。最成功的方法直接从原始输入进行训练，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设计特征更好的表示 [11]。这些成功激励了我们在强化学习中的方法。我们的目标是将强化学习算法与直接操作 RGB 图像的深度神经网络连接起来，并通过使用随机梯度更新高效处理训练数据。

Tesauro 的 TD-Gammon 架构为这种方法提供了一个起点。该架构通过直接从算法与环境交互（或在双陆棋的情况下通过自我对弈）中提取的策略经验样本  $s_t, a_t, r_t, s_{t+1}, a_{t+1}$ ，更新估计价值函数的网络参数。由于这种方法在 20 年前就已经能够超越最优秀的人类双陆棋选手，人们自然会思考，二十年的硬件进步，加上现代的深度神经网络架构和可扩展的强化学习算法，是否能够带来显著的进展。

与 TD-Gammon 及类似的在线方法不同，我们采用了一种称为 *experience replay* [13] 的技术，其中我们在每个时间步存储代理的经验  $e_t = (s_t, a_t, r_t, s_{t+1})$ ，汇集到数据集  $\mathcal{D} = e_1, \dots, e_N$  中，经过多轮次累积形成一个 *replay memory*。在算法的内部循环中，我们对从存储样本池中随机抽取的经验样本  $e \sim \mathcal{D}$  应用 Q 学习更新或小批量更新。执行经验回放后，代理根据  $\epsilon$ -贪婪策略选择并执行一个动作。由于将任意长度的历史作为神经网络的输入可能较为困难，我们的 Q 函数转而作用于由函数  $\phi$  生成的固定长度历史表示上。我们将完整算法命名为 *deep Q-learning*，其详细步骤见算法 1。

这种方法相较于标准的在线 Q 学习 [23] 具有多个优势。首先，每一步的经验都可能用于多次权重更新，从而提高了数据效率。

---

**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  $\epsilon$  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
```

---

其次，直接从连续样本中学习效率低下，因为样本之间存在强相关性；随机化样本可以打破这些相关性，从而减少更新的方差。第三，在策略学习时，当前参数决定了接下来用于训练参数的数据样本。例如，如果最大化动作是向左移动，那么训练样本将主要由左侧的样本主导；如果最大化动作随后切换到右侧，训练分布也会随之切换。不难看出，不良的反馈循环可能会出现，参数可能会陷入较差的局部最小值，甚至灾难性地发散[25]。通过使用经验回放，行为分布会在其许多先前状态上取平均，从而平滑学习过程，避免参数的振荡或发散。需要注意的是，在使用经验回放学习时，必须进行离策略学习（因为我们当前的参数与生成样本时使用的参数不同），这促使了Q-learning的选择。

在实践中，我们的算法仅存储最近的 $N$ 个经验元组于回放记忆中，并在执行更新时从 $\mathcal{D}$ 中均匀随机采样。这种方法在某些方面存在局限性，因为记忆缓冲区不会区分重要的转移，并且由于有限的内存大小 $N$ ，总是用最近的转移覆盖旧的。同样，均匀采样赋予回放记忆中所有转移同等的重要性。一种更为复杂的采样策略可能会强调那些我们能从中学习最多的转移，类似于优先扫描[17]的做法。

#### 4.1 预处理与模型架构

直接处理原始的Atari帧（即 $210 \times 160$ 像素、128色调色板的图像）可能在计算上要求较高，因此我们采用了一个基本的预处理步骤，旨在降低输入维度。原始帧首先通过将其RGB表示转换为灰度，并下采样为 $110 \times 84$ 图像来进行预处理。最终的输入表示是通过裁剪图像中大致捕捉到游戏区域的 $84 \times 84$ 区域获得的。最终裁剪阶段之所以必要，是因为我们使用了来自[11]的GPU实现的2D卷积，该实现期望输入为方形。对于本文中的实验，算法1中的函数 $\phi$ 将此预处理应用于历史记录中的最后4帧，并将它们堆叠以生成 $Q$ 函数的输入。

有多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-动作对映射到它们的 $Q$ 值的标量估计，因此一些先前的方法[20, 12]已将历史和动作作用作神经网络的输入。这种架构的主要缺点是，需要单独的前向传递来计算每个动作的 $Q$ 值，导致成本随动作数量线性增长。我们则采用了一种架构，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下各个动作的预测 $Q$ 值。这种架构的主要优点是能够在给定状态下仅通过一次前向传递计算所有可能动作的 $Q$ 值。

我们现在描述用于所有七款Atari游戏的精确架构。神经网络的输入是由 $\phi$ 生成的 $84 \times 84 \times 4$ 图像。第一个隐藏层使用16个 $8 \times 8$ 的滤波器，步幅为4，对输入图像进行卷积，并应用整流非线性[10, 18]。第二个隐藏层使用32个 $4 \times 4$ 的滤波器，步幅为2，同样跟随一个整流非线性。最后的隐藏层是全连接的，由256个整流单元组成。输出层是一个全连接的线性层，每个有效动作对应一个输出。在我们考虑的游戏里，有效动作的数量在4到18之间变化。我们将我们的方法训练的卷积网络称为深度Q网络（DQN）。

## 5 实验

迄今为止，我们已在七款流行的ATARI游戏上进行了实验——Beam Rider、Breakout、Enduro、Pong、Q\*bert、Seaquest、Space Invaders。我们在所有七款游戏中使用了相同的网络架构、学习算法和超参数设置，这表明我们的方法足够稳健，能够在无需融入游戏特定信息的情况下适用于多种游戏。虽然我们在真实且未修改的游戏中评估了我们的智能体，但在训练期间对游戏的奖励结构做了一项调整。鉴于各游戏间的得分尺度差异巨大，我们将所有正面奖励固定为1，所有负面奖励固定为-1，保持0奖励不变。通过这种方式裁剪奖励，限制了误差导数的规模，使得在多个游戏间使用相同学习率更为便捷。同时，这也可能影响我们智能体的表现，因为它无法区分不同大小的奖励。

在这些实验中，我们使用了RMSProp算法，并采用了大小为32的小批量。训练期间的行为策略是 $\epsilon$ -贪婪策略，其中 $\epsilon$ 在前一百万帧中从1线性退火到0.1，之后固定为0.1。我们总共训练了一千万帧，并使用了一百万帧的最近回放记忆。

遵循之前玩Atari游戏的方法，我们也采用了一种简单的跳帧技术[3]。更准确地说，智能体每隔 $k^{th}$ 帧观察并选择一次动作，而不是每一帧都选择，跳过的帧则重复其上一个动作。由于模拟器前进一步所需的计算量远小于智能体选择动作的计算量，这项技术使得智能体能够在不显著增加运行时间的情况下，大约多玩 $k$ 倍的游戏。除了《太空入侵者》外，我们对所有游戏都使用 $k=4$ ，因为在《太空入侵者》中，我们发现使用 $k=4$ 会导致激光因闪烁周期而不可见。为了使激光可见，我们使用了 $k=3$ ，这一变化是所有游戏中超参数值唯一的差异。

### 5.1 训练与稳定性

在监督学习中，人们可以通过在训练集和验证集上评估模型来轻松跟踪模型在训练期间的表现。然而，在强化学习中，准确评估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，是智能体在一局或一场游戏中获得的总奖励在若干游戏中的平均值，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，因为策略权重的微小变化可能导致策略访问的状态分布发生巨大变化。图2中最左侧的两个图表展示了在Seaquest和Breakout游戏中，平均总奖励在训练期间的变化情况。这两个平均奖励图表确实相当嘈杂，给人一种学习算法没有稳步进展的印象。另一个更稳定的指标是策略的估计动作值函数 $Q$ ，它提供了智能体从任何给定状态遵循其策略所能获得的折扣奖励的估计值。我们在训练开始前通过运行随机策略收集一组固定状态，并跟踪这些状态的最大预测值<sup>2</sup>的平均值 $Q$ 。图2中最右侧的两个图表显示，平均预测值 $Q$ 比智能体获得的平均总奖励增长得更加平滑，并且在其他五个游戏中绘制相同的指标也产生了类似的平滑曲线。除了训练期间看到预测值 $Q$ 相对平滑的改进外，我们在任何实验中都没有遇到任何发散问题。这表明，尽管缺乏任何理论上的收敛保证，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。

<sup>2</sup>The maximum for each state is taken over the possible actions.

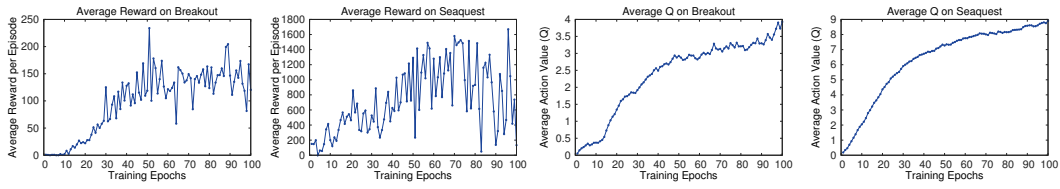


图2：左侧两图分别展示了训练期间Breakout和Seaquest上每回合的平均奖励。统计数据是通过运行一个 $\epsilon$ -贪婪策略，其中 $\epsilon = 0.05$ ，进行10000步计算得出的。右侧两图则分别展示了Breakout和Seaquest上保留状态集的平均最大预测动作值。一个周期对应50000次小批量权重更新，或大约30分钟的训练时间。

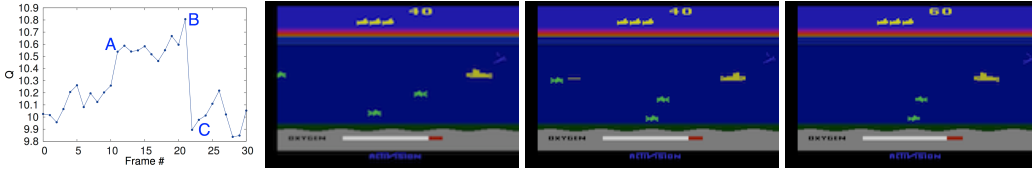


图3：最左侧的图表展示了游戏《海战》中一段30帧片段的预测价值函数 $\{v^*\}$ 。三个屏幕截图分别对应标记为A、B和C的帧。

## 5.2 价值函数可视化

图3展示了在游戏《海战》中学到的价值函数的可视化。图中显示，当敌人在屏幕左侧出现时（点A），预测值会跳跃上升。随后，代理向敌人发射鱼雷，当鱼雷即将击中敌人时，预测值达到峰值（点B）。最后，敌人消失后，价值大致回落到其原始值（点C）。图3表明，我们的方法能够学习价值函数在相当复杂的事件序列中如何演变。

## 5.3 主要评估

我们将自己的结果与强化学习文献中表现最佳的方法进行了比较[3, 4]。标记为Sarsa的方法使用Sarsa算法在多个为Atari任务手工设计的特征集上学习线性策略，并报告了表现最佳的特征集的得分[3]。Contingency采用了与Sarsa相同的基本方法，但通过增加一个学习到的表示来增强特征集，该表示涵盖了屏幕上由代理控制的部分[4]。需要注意的是，这两种方法都通过使用背景减除技术并将128种颜色中的每一种视为独立通道，融入了关于视觉问题的显著先验知识。由于许多Atari游戏为每种对象类型使用一种独特的颜色，将每种颜色视为独立通道可能类似于生成一个独立的二进制图，编码每种对象类型的存在。相比之下，我们的代理仅接收原始的RGB截图作为输入，并且必须*learn*自行检测对象。

除了学习到的智能体外，我们还报告了一位专家级人类玩家和一种随机均匀选择动作的策略的得分。人类表现是在每款游戏大约玩两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]中的得分。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，并报告通过运行 $\epsilon$ -贪婪策略在固定步数内获得的平均得分，其中 $\epsilon = 0.05$ 。表1的前五行显示了所有游戏的每款游戏平均得分。我们的方法（标记为DQN）在所有七款游戏中均以显著优势优于其他学习方法，尽管几乎没有融入任何关于输入的先验知识。

我们还在表1的最后三行中包含了与[8]中进化策略搜索方法的比较。我们报告了该方法的两组结果。HNeat最佳分数反映了通过使用手动设计的对象检测算法获得的结果，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	<b>4092</b>	<b>168</b>	<b>470</b>	<b>20</b>	<b>1952</b>	<b>1705</b>	<b>581</b>
Human	7456	31	368	-3	18900	28010	3690
HNeat Best [8]	3616	52	106	19	1800	920	<b>1720</b>
HNeat Pixel [8]	1332	4	91	-16	1325	800	1145
DQN Best	<b>5184</b>	<b>225</b>	<b>661</b>	<b>21</b>	<b>4500</b>	<b>1740</b>	1075

表1：上表比较了通过运行 $\epsilon$ -贪婪策略（ $\epsilon = 0.05$ ）在固定步数下各种学习方法的平均总奖励。下表报告了HNeat和DQN在单次最佳表现回合中的结果。HNeat生成确定性策略，总是获得相同的分数，而DQN使用了 $\epsilon$ -贪婪策略（ $\epsilon = 0.05$ ）。

雅达利屏幕上的对象类型。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，该表示在每个通道上代表一个对象标签图。这种方法严重依赖于找到一个代表成功利用的确定性状态序列。以这种方式学习的策略不太可能推广到随机扰动；因此，该算法仅在得分最高的单次游戏中进行评估。相比之下，我们的算法在 $\epsilon$ -贪婪控制序列上进行评估，因此必须在各种可能的情况下进行推广。然而，我们展示出，在除《太空入侵者》之外的所有游戏中，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）都实现了更好的性能。

最后，我们展示了我们的方法在《打砖块》、《耐力赛》和《乒乓球》上超越了人类专家玩家的表现，并在《光束骑士》上接近了人类水平。而在《Q\*bert》、《海底大战》、《太空侵略者》这些游戏中，我们距离人类表现还有较大差距，这些游戏更具挑战性，因为它们要求网络找到一种能在长时间跨度内有效的策略。

## 6 结论

本文介绍了一种新的深度学习模型用于强化学习，并展示了其仅使用原始像素作为输入即可掌握Atari 2600电脑游戏中复杂控制策略的能力。我们还提出了一种在线Q学习的变体，该变体将随机小批量更新与经验回放记忆相结合，以简化深度网络在强化学习中的训练。我们的方法在测试的七款游戏中的六款上取得了最先进的结果，且无需调整架构或超参数。

## 参考文献

- [1] Leemon Baird. 残差算法：使用函数逼近的强化学习。在 *Proceedings of the 12th International Conference on Machine Learning (ICML 1995)* 中，第30-37页。Morgan Kaufmann, 1995年。[2] Marc Bellemare, Joel Veness, 和 Michael Bowling. 基于草图的线性值函数逼近。在 *Advances in Neural Information Processing Systems* 25 中，第222-2230页，2012年。[3] Marc G Bellemare, Yavar Naddaf, Joel Veness, 和 Michael Bowling. 街机学习环境：通用代理的评估平台。 *Journal of Artificial Intelligence Research*, 47:253-279, 2013年。[4] Marc G Bellemare, Joel Veness, 和 Michael Bowling. 使用Atari 2600游戏研究应急意识。在 *AAAI* 中，2012年。[5] Marc G. Bellemare, Joel Veness, 和 Michael Bowling. 递归分解环境的贝叶斯学习。在 *Proceedings of the Thirtieth International Conference on Machine Learning (ICML 2013)* 中，第1211-1219页，2013年。



[6] George E. Dahl, Dong Yu, Li Deng, 和 Alex Acero. 用于大词汇量语音识别的上下文相关预训练深度神经网络。 *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1): 30 – 42, 2012年1月。

[7] Alex Graves, Abdel-rahman Mohamed, 和 Geoffrey E. Hinton. 使用深度循环神经网络的语音识别。于 *Proc. ICASSP*, 2013年。

[8] Matthew Hausknecht, Ris to Miikkulainen, 和 Peter Stone. 一种通用的Atari游戏玩法的神经进化方法。2013年。

[9] Nicolas Heess, David Silver, 和 Yee Whye Teh. 基于能量策略的演员-评论家强化学习。于 *European Workshop on Reinforcement Learning*, 第43页, 2012年。

[10] Kevin Jarrett, Koray Kavukcuoglu, MarcAurelio Ranzato, 和 Yann LeCun. 什么是物体识别的最佳多阶段架构? 于 *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, 第2146 – 2153页。IEEE, 2009年。

[11] Alex Krizhevsky, Ilya Sutskever, 和 Geoff Hinton. 使用深度卷积神经网络的ImageNet分类。于 *Advances in Neural Information Processing Systems 25*, 第1106 – 1114页, 2012年。

[12] Sascha Lange 和 Martin Riedmiller. 强化学习中的深度自编码神经网络。于 *Neural Networks (IJCNN), The 2010 International Joint Conference on*, 第1 – 8页。IEEE, 2010年。

[13] Long-Ji Lin. 使用神经网络的机器人强化学习。技术报告, D TIC Document, 1993年。

[14] Hamid Maei, Csaba Szepesvari, Shalabh Bhatnagar, Doina Prec up, David Silver, 和 Rich Sutton. 任意平滑函数逼近的收敛时间差分学习。于 *Advances in Neural Information Processing Systems 22*, 第1204 – 1212页, 2009年。

[15] Ham id Maei, Csaba Szepesvári, Shalabh Bhatnagar, 和 Richard S. Sutton. 迈向基于函数逼近的非策略学习控制。于 *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, 第719 – 726页, 2010年。

[16] Volodymyr Mnih. *Machine Learning for Aerial Image Labeling*. 博士论文, 多伦多大学, 2013年。

[17] Andrew Moore 和 Chris Atkeson. 优先扫描: 减少数据和实时需求的强化学习。 *Machine Learning*, 13:103 – 130, 1993年。

[18] Vinod Nair 和 Geoffrey E Hinton. 修正线性单元改进受限玻尔兹曼机。于 *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, 第807 – 814页, 2010年。

[19] Jordan B. Pollack 和 Alan D. Blair. 为什么TD-Gammon成功了。于 *Advances in Neural Information Processing Systems 9*, 第10 – 16页, 1996年。

[20] Martin Riedmiller. 神经拟合Q迭代——一种数据高效的神经强化学习方法的初步经验。于 *Machine Learning: ECML 2005*, 第317 – 328页。Springer, 2005年。

[21] Brian Sallans 和 Geoffrey E. Hinton. 基于因子化状态和动作的强化学习。 *Journal of Machine Learning Research*, 5:1063 – 1088, 2004年。

[22] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, 和 Yann LeCun. 无监督多阶段特征学习的行人检测。于 *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR 2013)*。IEEE, 2013年。

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

[24] Gerald Tesauro. 时间差分学习与TD-Gammon。 *Communications of the ACM*, 38(3):58 – 68, 1995年。

[25] John N Tsitsiklis 和 Benjamin Van Roy. 时间差分学习与函数逼近的分析。 *Automatic Control, IEEE Transactions on*, 42(5):674 – 690, 1997年。

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