# 用深度强化学习玩转Atari

**摘要**

我们提出了第一个使用强化学习、可直接从高维感官输入中成功学习到控制策略的深度学习模型。 这模型是一个卷积神经网络，用 Q-learning 的变体训练，其输入是原始像素，其输出是估计未来奖励的值函数。 我们将我们的方法应用于来自 Arcade 学习环境的七款 Atari 2600 游戏，而没有调整架构或学习算法。 我们发现它在六场比赛中优于所有以前的方法并超过其中三个的人类专家。

**1、简介(详细版摘要)**

直接从视觉和语音等高维感官输入中进行限制因素的学习是强化学习 (RL) 的长期挑战之一。在这些应用领域上，大多数成功的 RL 应用程序都依赖于手工制作的特征与线性值函数或策略表示相结合。显然，此类系统的性能在很大程度上依赖于特征表示的质量。

深度学习的最新进展使得从原始传感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] 方面取得突破。这些方法利用了一系列神经网络架构，包括卷积网络、多层感知器、受限玻尔兹曼机和循环神经网络，并利用了监督和无监督学习。**要求类似的技术是否也有利于具有感官数据的 RL 似乎很自然。**

然而，从深度学习的角度来看，强化学习提出了一些挑战。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记的训练数据。另一方面，RL 算法必须能够从标量奖励中学习经常稀疏、嘈杂和延迟的信号。行动和由此产生的奖励之间的延迟，这可能有数千个时间步长，在监督学习中，直接与发现的输入和目标之间的关联相比，这似乎特别令人生畏。另一个问题是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 RL 中，数据分布会随着算法学习新行为而发生变化，这对于深度学习假设固定基础分布的方法来说，可能是个问题。

**这篇论文证明了卷积神经网络可以克服这些挑战，在复杂的强化学习环境中，成功学习来自原始视频数据的控制策略。** 网络是使用 Q-learning [26] 算法的变体进行训练，使用随机梯度下降进行更新权重。 为了缓解相关数据和非平稳分布的问题，我们使用一种经验重放机制 [13]，它随机采样先前的转换，从而平滑了许多过去行为的训练分布。



图1：五款 Atari 2600 游戏的屏幕截图：（从左到右）Pong,Breakout,Space Invaders,Seaquest,BeamRider

我们将我们的方法应用于在街机学习环境 (ALE) [3] 中实现的一系列 Atari 2600 游戏。 Atari 2600 是一个具有挑战性的 RL 测试平台，它为代理提供高维视觉输入（60Hz 的 210 × 160 RGB 视频）和一组多样化且有趣的任务，对于人类玩家来说，这被设计的很困难。我们的目标是创建一个单一的神经网络代理，它能够成功地学会玩尽可能多的游戏。该网络没有提供任何特定于游戏的信息或手工设计的视觉特征，也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，就像人类玩家一样。此外，用于训练的网络架构和所有超参数在整个游戏中保持不变。迄今为止，在我们尝试过的七场比赛中的六场比赛中，网络的表现优于所有以前的强化学习算法，并且在他们三人上超越了一位专业的人类玩家。图 1 提供了来自五个用于训练的游戏样本截图。

**2、背景**

我们考虑代理网络与环境  交互的任务，在这种情况下是 Atari 模拟器，在一系列动作、观察和奖励中。 在每个时间步长，代理从一组合法的游戏动作中选择一个动作，A = {1, . . . ，K}。 动作被传递给模拟器并修改其内部状态和游戏分数。 一般来说，E 可能是随机的。 代理不会观察到模拟器的内部状态； 相反，它从模拟器观察图像 xt ∈ Rd，它是代表当前屏幕的原始像素值向量。 此外，它还会收到一个表示游戏分数变化的奖励 rt。 请注意，通常游戏分数可能取决于整个先前的动作和观察顺序； 可能只有在经过数千个时间步之后才能收到有关某个动作的反馈。

由于代理只观察当前屏幕的图像，任务被部分观察，许多仿真器状态在感知上是混叠的，即不可能完全理解当前情况仅从当前屏幕 xt。 因此，我们考虑动作和观察序列，st = x1, a1, x2, ..., at−1, xt，并学习依赖于这些序列的游戏策略。 假设仿真器中的所有序列都在有限数量的时间步长内终止。 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，只需使用完整的序列 st 作为时间 t 的状态表示。

代理的目标是通过以最大化未来奖励的方式选择动作来与模拟器交互。 我们做出标准假设，即未来的奖励在每个时间步长上折现一个因子 γ，并将时间 t 的未来折现回报定义为 Rt = PTt0=t γt0−trt0 ，其中 T 是游戏的时间步长 终止。 我们定义了最优动作值函数 Q∗(s, a)作为遵循任何策略可实现的最大预期回报，在看到一些序列然后采取一些行动 a 之后，Q∗(s, a) = maxπ E [Rt|st = s, at = a, π]，其中 π 是 a 策略映射序列到动作（或动作分布）。

最优动作值函数遵循一个重要的恒等式，称为贝尔曼方程。 这是基于以下直觉：如果对于所有可能的动作 a0，下一时间步的序列 s0 的最优值 Q∗(s0, a0) 是已知的，那么最优策略是选择动作 a0 最大化预期 r + γQ∗(s0, a0) 的值，



许多强化学习算法背后的基本思想是估计动作价值函数，通过使用贝尔曼方程作为迭代更新，Qi+1(s, a) =E [r + γ maxa0 Qi(s0, a0)|s ， 一种]。 这样的值迭代算法收敛到最优动作值函数，Qi → Q∗as i → ∞ [23]。 在实践中，这种基本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，通常使用函数逼近器来估计动作值函数，Q(s, a; θ) ≈ Q∗(s, a)。 在强化学习社区中，这通常是一个线性函数近似器，但有时会使用非线性函数近似器，例如神经网络网络。 我们将权重为 θ 的神经网络函数逼近器称为 Q 网络。 一种Q-network 可以通过最小化一系列损失函数 Li(θi) 来训练迭代 i，



其中 yi = Es0∼E [r + γ maxa0 Q(s0, a0; θi−1)|s, a] 是迭代 i 的目标，ρ(s, a) 是 a序列 s 和动作 a 的概率分布，我们称之为行为分布。优化损失函数时，上次迭代 θi−1 的参数保持固定力 (θi)。 请注意，目标取决于网络权重； 这与使用的目标相反用于监督学习，在学习开始之前是固定的。 微分损失函数关于权重，我们得出以下梯度，



与计算上述梯度中的全部期望不同，通过随机梯度下降来优化损失函数通常在计算上是有利的。 如果权重在之后更新每个时间步长，期望被行为分布中的单个样本替换ρ 和模拟器 E，然后我们到达熟悉的 Q 学习算法 [26]。

请注意，该算法是无模型的：它直接使用来自模拟器 E 的样本来解决强化学习任务，而没有明确构建对 E 的估计。它也是离策略的：它学习贪婪策略 a = maxa Q(s, a; θ)，同时遵循行为分布确保对状态空间的充分探索。 在实践中，行为分布通常由遵循贪婪策略的 -greedy 策略选择，概率为 1 - 并选择一个具有概率的随机动作。

1. **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来学习的西洋双陆棋程序，并达到了超人类水平的发挥[24]。TD-gammon 使用了一种类似的无模型强化学习算法到 Q-learning，并使用具有一个隐藏层的多层感知器近似值函数层1。

然而，早期尝试跟进 TD-gammon，包括将相同的方法应用于国际象棋、围棋和跳棋不太成功。 这导致人们普遍认为 TD-gammon方法是一个特例，只适用于西洋双陆棋，也许是因为掷骰子有助于探索状态空间，并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（例如 Q 学习）与非线性函数逼近器 [25] 或实际上与离策略学习 [1] 相结合，可以导致 Q 网络发散。 随后，强化学习中的大部分工作都集中在具有更好收敛性保证的线性函数逼近器上 [25]。

【事实上，TD-Gammon 近似的是状态值函数 V(s) 而不是动作值函数Q(s, a)，并直接从自我博弈中学习到on-policy】

最近，人们对将深度学习与强化相结合产生了兴趣学习。 深度神经网络已被用于估计环境 E； 受限玻尔兹曼机器已被用于估计价值函数[21]； 或政策[9]。 除此之外Q-learning 的发散问题已部分通过梯度时间差异解决方法。 在评估具有非线性的固定策略时，这些方法被证明是收敛的函数逼近器[14]； 或者在学习线性函数近似的控制策略时使用 Q-learning 的受限变体 [15]。 但是，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优化方程2中损失函数的序列，使用RPROP算法更新Q 网络的参数。但是，它使用具有计算成本的批量更新每次迭代与数据集的大小成正比，而我们考虑随机梯度每次迭代具有较低的恒定成本并扩展到大型数据集的更新。 NFQ也被使用纯视觉输入成功应用于简单的现实世界控制任务，首先使用深度自动编码器学习任务的低维表示，然后将 NFQ 应用于此代表[12]。相比之下，我们的方法直接端到端地应用强化学习来自视觉输入；因此，它可以学习与区分直接相关的特征行动价值。 Q-learning 之前也结合了经验回放和简单的神经网络 [13]，但再次从低维状态而不是原始视觉输入开始。

1. 介绍了使用 Atari 2600 模拟器作为强化学习平台，他应用具有线性函数逼近和泛型的标准强化学习算法视觉特征。 随后，通过使用更多的特征改进了结果，并且使用拔河哈希将特征随机投影到低维空间 [2]。 这HyperNEAT 进化架构 [8] 也已应用于 Atari 平台，在那里它用于进化（单独，对于每个不同的游戏）一个代表该策略的神经网络游戏。 当使用模拟器的重置工具针对确定性序列反复训练时，这些策略能够利用几个 Atari 游戏中的设计缺陷。

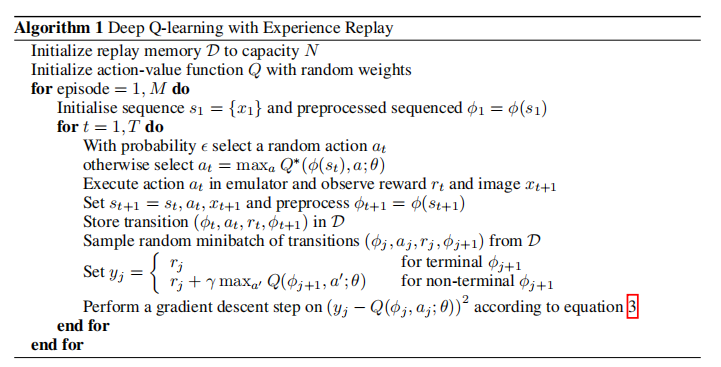
**4、深度强化学习**

计算机视觉和语音识别的最新突破依赖于非常大的训练集上的高效的训练的深度神经网络。 最成功的方法是直接训练的从原始输入，使用基于随机梯度下降的轻量级更新。 通过输入足够的数据进入深度神经网络，通常可能比手工处理特征有更好的学习效果[11]。 这些成功激励了我们采用强化学习的方法。 我们的目标是将强化学习算法连接到运行的深度神经网络，它直接在 RGB 图像上，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。 该架构直接从 经验策略样本中，更新估计价值函数的网络的参数，st、at、rt、st+1、at+1，来自算法与环境的交互（或在双陆棋的情况下通过自我对弈）。 由于这种方法能够胜过20 年前人类最好的西洋双陆棋选手，很自然地想知道 20年硬件的改进，加上现代深度神经网络架构和可扩展的强化学习算法可能会产生重大进展。

与 TD-Gammon 和类似的在线方法相比，我们使用了一种称为经验重放的技术 [13]，我们在每个时间步存储代理的经验，et = (st, at, rt, st+1)在数据集 D = e1, ..., eN 中，将许多情节汇集到重播记忆中。 期间内在算法的循环中，我们将 Q-learning 更新或小批量更新应用于经验样本，e ∼ D，从存储的样本池中随机抽取。 进行体验回放后，代理根据 -greedy 策略选择并执行一个动作。 由于使用历史任意长度作为神经网络的输入可能很困难，我们的 Q 函数改为适用于固定由函数 φ 产生的历史的长度表示。 完整的算法，我们称之为 deep Q-learning，在算法 1 中给出。

与标准的在线 Q-learning [23] 相比，这种方法有几个优点。 首先，每一步经验可能用于许多权重更新，从而提高数据效率。



其次，由于强相关性，直接从连续样本中学习效率低下样品之间；随机化样本打破了这些相关性，因此减少了更新的差异。第三，在学习on-policy时，当前参数决定下一个训练参数的数据样本。例如，如果最大化操作是向左移动那么训练样本将以来自左侧的样本为主；如果最大化动作然后向右切换，那么训练分布也将切换。很容易看出如何可能会出现不需要的反馈循环，并且参数可能会陷入不良的局部最小值，或者甚至发生灾难性的分歧 [25]。通过使用经验回放，平均行为分布在许多以前的状态上，平滑学习并避免振荡或发散参数。注意，通过经验回放学习时，需要学习off-policy（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-learning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，并且样本执行更新时均匀地从 D 随机。 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总是用由于有限的内存大小 N，最近的转换。类似地，均匀采样给出了相等的重放记忆中所有转换的重要性。 更复杂的抽样策略可能强调我们可以从中学到最多的转换，类似于优先扫描 [17]。

**4.1 预处理和模型架构**

直接使用原始 Atari 帧，即 210 × 160 像素图像和 128 色板，可能对计算要求很高，所以我们应用了一个基本的预处理步骤，旨在减少输入维度。原始帧通过首先转换它们的 RGB 表示进行预处理灰度并将其下采样为 110×84 图像。最终输入表示由裁剪图像的 84 × 84 区域，大致捕获播放区域。最后的裁剪仅需要 stage 是因为我们使用了 [11] 中的 2D 卷积的 GPU 实现，其中期望平方输入。对于本文中的实验，算法 1 中的函数 φ 应用了这个对历史的最后 4 帧进行预处理并将它们堆叠起来以产生 Q 函数的输入。

有几种使用神经网络参数化 Q 的可能方法。由于 Q 将历史动作对映射到其 Q 值的标量估计，因此历史和动作已被用作输入通过一些先前的方法 [20, 12] 连接到神经网络。这种类型的主要缺点架构的特点是需要单独的前向传递来计算每个动作的 Q 值，导致成本与操作数量成线性比例。我们改为使用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。输出对应于个体的预测 Q 值输入状态的动作。这种架构的主要优点是能够计算给定状态下所有可能动作的 Q 值，只有一次前向通过网络。

我们现在描述用于所有七款 Atari 游戏的确切架构。 神经网络的输入网络由 φ 生成的 84 × 84 × 4 图像组成。 第一个隐藏层卷积 16 8 × 8输入图像的步长为 4 的滤波器并应用整流器非线性 [10, 18]。 第二隐藏层卷积 32 个 4 × 4 的滤波器，步长为 2，再次跟随非线性整流器。 这最后一个隐藏层是全连接的，由 256 个整流单元组成。 输出层是一个全连接的线性层，每个有效动作都有一个输出。 有效动作的数量各不相同在我们考虑的游戏中，4 到 18 之间。 我们指的是用我们的训练的卷积网络作为深度 Q 网络 (DQN) 的方法。

**5、试验**

到目前为止，我们已经对七款流行的 ATARI 游戏进行了实验——Beam Rider、Breakout、Enduro、Pong、Q\*bert、Seaquest、太空侵略者。我们使用相同的网络架构，学习所有七场比赛的算法和超参数设置，表明我们的方法是稳健的足以在不包含游戏特定信息的情况下处理各种游戏。虽然我们在真实和未修改的游戏中评估我们的代理，我们对奖励结构进行了一项更改仅在训练期间的比赛。由于分数的规模因游戏而异，我们将所有正奖励固定为 1，将所有负奖励固定为 -1，保持 0 奖励不变。以这种方式削减奖励限制了误差导数的规模，并使其更容易在多个游戏中使用相同的学习率。同时，它可能会影响性能我们的代理，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 RMSProp 算法和大小为 32 的小批量。训练期间的策略是 -greedy，在前一百万个中从 1 线性退火到 0.1帧，此后固定为 0.1。我们总共训练了 1000 万帧并使用了回放最近一百万帧的内存。

按照之前玩 Atari 游戏的方法，我们还使用了一种简单的跳帧技术 [3]。更准确地说，代理查看并选择每 k 个动作第一个帧而不是每个帧，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个步骤比让代理选择一个动作需要更少的计算，这种技术允许代理可以玩大约 k 倍的游戏而不会显着增加运行时间。我们用对于除 Space Invaders 之外的所有游戏，k = 4，我们注意到使用 k = 4 会使激光由于它们眨眼的时间段而看不见。我们使用 k = 3 使激光可见，并且这一变化是任何游戏之间超参数值的唯一差异。

**5.1 训练和稳定性**

在监督学习中，人们可以通过评估在训练期间轻松跟踪模型的性能它在训练和验证集上。然而，在强化学习中，准确评估代理在训练期间的进步可能具有挑战性。由于我们的评估指标，如建议由 [3] 计算，是智能体在一集或游戏中收集的总奖励在多个游戏，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会导致分布的巨大变化陈述政策访问。图 2 中最左边的两个图显示了平均总奖励如何演变在游戏 Seaquest和 Breakout 的训练期间。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步进展的印象。其他，更稳定的度量是策略的估计动作价值函数 Q，它提供了对通过从任何给定状态遵循其政策，代理可以获得多少折扣奖励。我们通过在训练开始之前运行随机策略来收集一组固定的状态并跟踪平均值这些状态的最大 2 预测 Q 值。图 2 中最右边的两个图显示平均预测 Q 的增加比代理获得的平均总奖励要平滑得多，并且在其他五场比赛中绘制相同的指标会产生类似的平滑曲线。此外在训练期间看到对预测 Q 的相对平稳的改进，我们没有遇到任何我们任何实验中的分歧问题。这表明，尽管缺乏任何理论收敛保证，我们的方法能够使用强化训练大型神经网络以稳定的方式学习信号和随机梯度下降。

【2每个状态的最大值被用于可能的操作。】

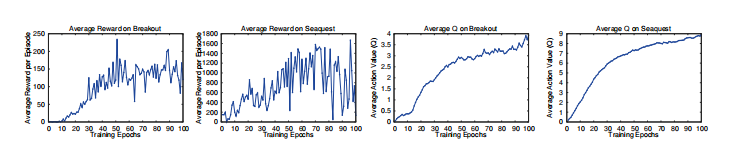


图 2：左侧的两个图显示了 Breakout 和 Seaquest 每集的平均奖励分别在训练时。统计数据是通过运行 10000 步 = 0.05 的 -greedy 策略来计算的。 右侧的两个图显示了平均最大预测动作值分别在 Breakout 和 Seaquest 上保留的一组状态。 1个epoch对应50000 minibatch 权重更新或大约 30 分钟的训练时间。



图 3：最左边的图显示了游戏 30 帧片段的预测值函数海求。三个屏幕截图分别对应于标记为 A、B 和 C 的帧。

**5.2 可视化值函数**

图 3 显示了游戏 Seaquest 上学习值函数的可视化。 如图所示在屏幕左侧（A点）出现敌人后，预测值会发生跳跃。 中介然后向敌人发射鱼雷，当鱼雷即将击中敌人时，预测值达到峰值敌人（B 点）。 最后，在敌人消失后，该值大致下降到其原始值（C点）。 图 3 表明我们的方法能够学习价值函数如何演变一系列相当复杂的事件。

**5.3 主要评价**

我们将我们的结果与 RL 文献 [3, 4] 中表现最佳的方法进行比较。方法标记为 Sarsa 使用 Sarsa 算法在为 Atari 任务手工设计的几个不同特征集上学习线性策略，我们报告了最佳性能特征集的分数 [3]。应急使用与 Sarsa 相同的基本方法，但通过学习代理控制下的屏幕部分的表示 [4]。注意这两个这些方法通过使用背景减法并将 128 种颜色中的每一种都视为单独的通道，从而结合了有关视觉问题的重要先验知识。由于许多 Atari 游戏使用

每种类型的对象都有一种不同的颜色，将每种颜色视为单独的通道可以类似于生成一个单独的二进制映射，编码每个对象类型的存在。相比之下，我们的代理只接收原始 RGB 屏幕截图作为输入，并且必须学会自己检测对象。

除了学习到的代理之外，我们还报告了专家人类游戏玩家的分数和策略随机均匀地选择动作。人类表现是获得的中值奖励每个游戏玩了大约两个小时后。请注意，我们报告的人类得分要高得多比 Bellemare 等人的那些。 [3]。对于学习到的方法，我们遵循使用的评估策略在 Bellemare 等人中。 [3, 5] 并报告通过运行 -greedy 策略获得的平均分数= 0.05 固定步数。表1的前五行显示了每场比赛的平均得分在所有游戏中。我们的方法（标记为 DQN）明显优于其他学习方法尽管几乎没有包含有关输入的先验知识，但所有七场比赛的利润率。

我们还在最后三个中包含与 [8] 中的进化策略搜索方法的比较表 1 的行。我们报告了此方法 的两组结果。 HNeat 最佳得分反映了通过使用手动设计的对象检测器算法获得的结果，该算法输出位置和Atari 屏幕上的对象类型。 HNeat Pixel 分数是通过使用特殊的 8 色获得的Atari 模拟器的通道表示，表示每个通道的对象标签映射。这种方法在很大程度上依赖于找到代表成功的确定性状态序列开发。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集上进行评估。相比之下，我们的算法在 -greedy 控制序列上进行评估，因此必须在广泛的范围内推广各种可能的情况。尽管如此，我们表明在所有游戏中，除了《太空侵略者》，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）都取得了更好的成绩表现。

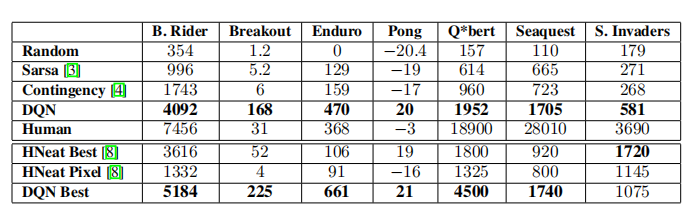


表 1：上表通过运行比较了各种学习方法的平均总回报对于固定的步数，具有 = 0.05 的 -greedy 策略。下表报告了以下结果HNeat 和 DQN 表现最好的一集。 HNeat 产生确定性的策略总是得到相同的分数，而 DQN 使用了一个 = 0.05 的 -greedy 策略。

最后，我们表明我们的方法比专家级人类玩家获得了更好的性能Breakout、Enduro 和 Pong，它在 Beam Rider 上的表现接近人类。游戏Q\*bert、Seaquest、Space Invaders，我们在这些方面与人类的表现相去甚远，它们更具挑战性，因为它们需要网络找到一种可以在很长一段时间内扩展的策略。

**六、结论**

本文介绍了一种新的用于强化学习的深度学习模型，并展示了其仅使用原始像素作为输入，掌握 Atari 2600 电脑游戏的困难控制策略的能力。 我们还提出了一种在线 Q 学习的变体，它将随机的、小批量的更新日期与经验重放记忆相结合，以简化 RL 深度网络的训练。 我们的方法在测试的七场比赛中，有六场取得了最先进的结果，没有调整架构或超参数。