中国科学技术大学
University of Science and Technology of China

《人工智能数学原理与算法》
第 7 章：强化学习

深度 Q 学习（DQN）

吉建民
jianmin@ustc.edu.cn

目录

目录

背景

- **Q 学习（Q-learning）**：是一种基于价值函数的强化学习算法，通过学习一个 Q 函数来表示在某一状态下采取某一动作所获得的最大期望回报。
  - Q-learning 的目标是找到最优策略，通过最大化每个状态的 Q 值来获得最优的行为策略。

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

其中，α 是学习率，γ 是折扣因子（衰减系数）。

  - 在强化学习领域，Q-learning 和 DQN 是两种非常重要且被广泛应用的算法。
  - **深度 Q 学习（Deep Q-Learning or Deep Q Network, DQN）**是 Q-learning 的深度学习版本，解决了Q-learning在处理高维状态空间时的难题。

什么是 DQN

- **深度 Q 学习（DQN）**：使用神经网络来逼近 Q 值函数，从而解决高维状态空间的强化学习问题。
  - 与传统的 Q 学习算法相比，DQN 不再使用表格来存储每个状态—动作对的 Q 值，而是通过一个深度神经网络（通常是卷积神经网络 CNN）来估计 Q 值。
  - DQN 通过结合经验回放（Experience Replay ）和目标网络（Target Network），成功地在多种复杂任务（如 Atari 游戏）中取得了显著的效果。

Mnih, V., Kavukcuoglu, K., Silver, D. *et al.* Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).

DQN 算法

A simple DQN Algorithm

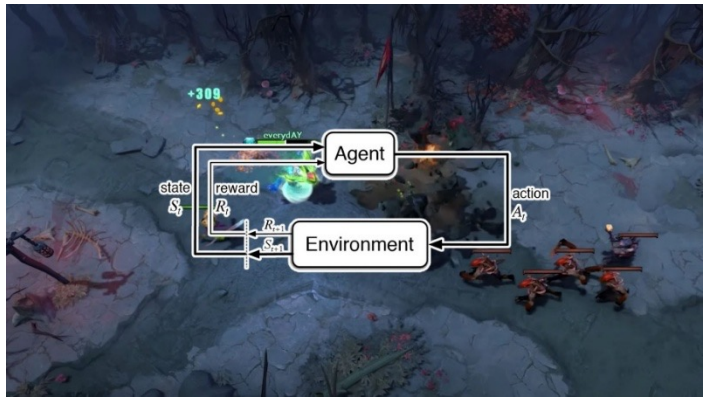
1. take action a_i in ENV,then insert (s_i, a_i, s_{i+1}, r_i) in Experience Replay
2. if Experience Replay have enough τ , then randomly sample (s_i, a_i, s_{i+1}, r_i) from Experience Replay
3. $target(s_i) = r_i + \gamma \max_{a'} Q_{\phi^-}(s_{i+1}, a')$
4. $\phi \leftarrow \phi + \alpha(target(s_i) - Q_\phi(s_i, a_i))\frac{\mathrm{d}Q_\phi}{\mathrm{d}\phi}$
5. every N step, let $\phi^- = \phi$
6. go to step 1

目录

DQN 应用

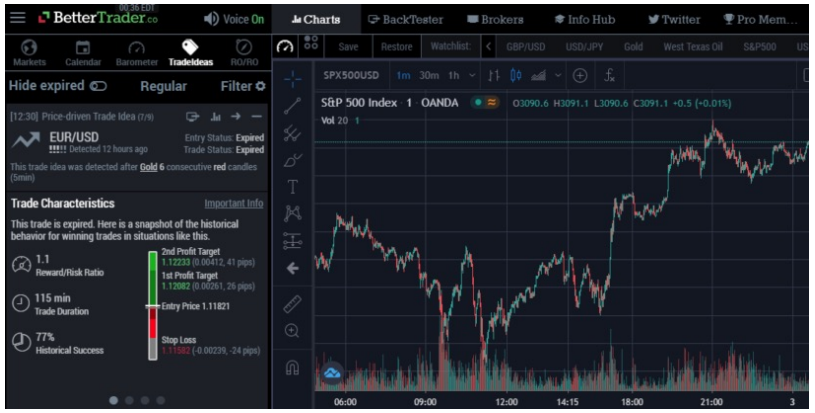
- 深度 Q 网络（DQN）目前广泛应用于游戏 AI、机器人控制中多智能体协同。
- 其改进版本（如 Double/Dueling DQN）进一步优化了策略稳定性，支撑多智能体协作、金融交易及智能交通信号优化等场景

游戏领域



DQN在Atari游戏中取得突破性进展

量化投资



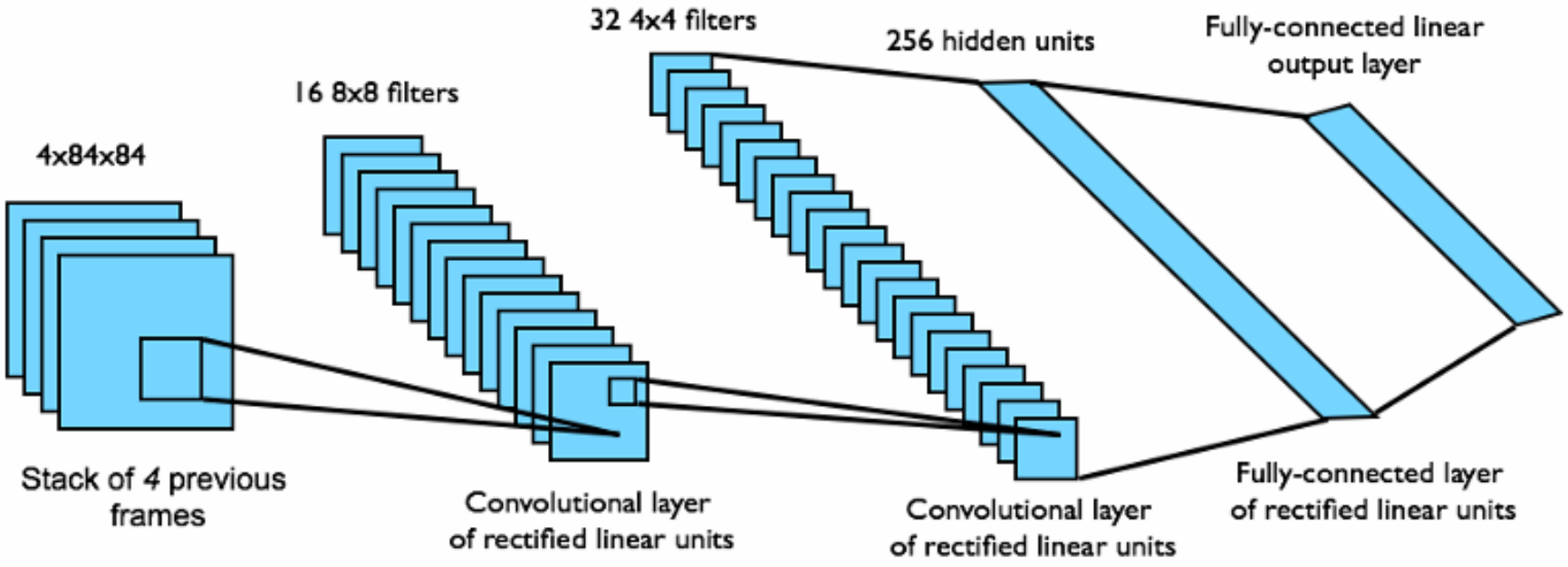
通过动态资产配置降低市场波动风险

智能机器人

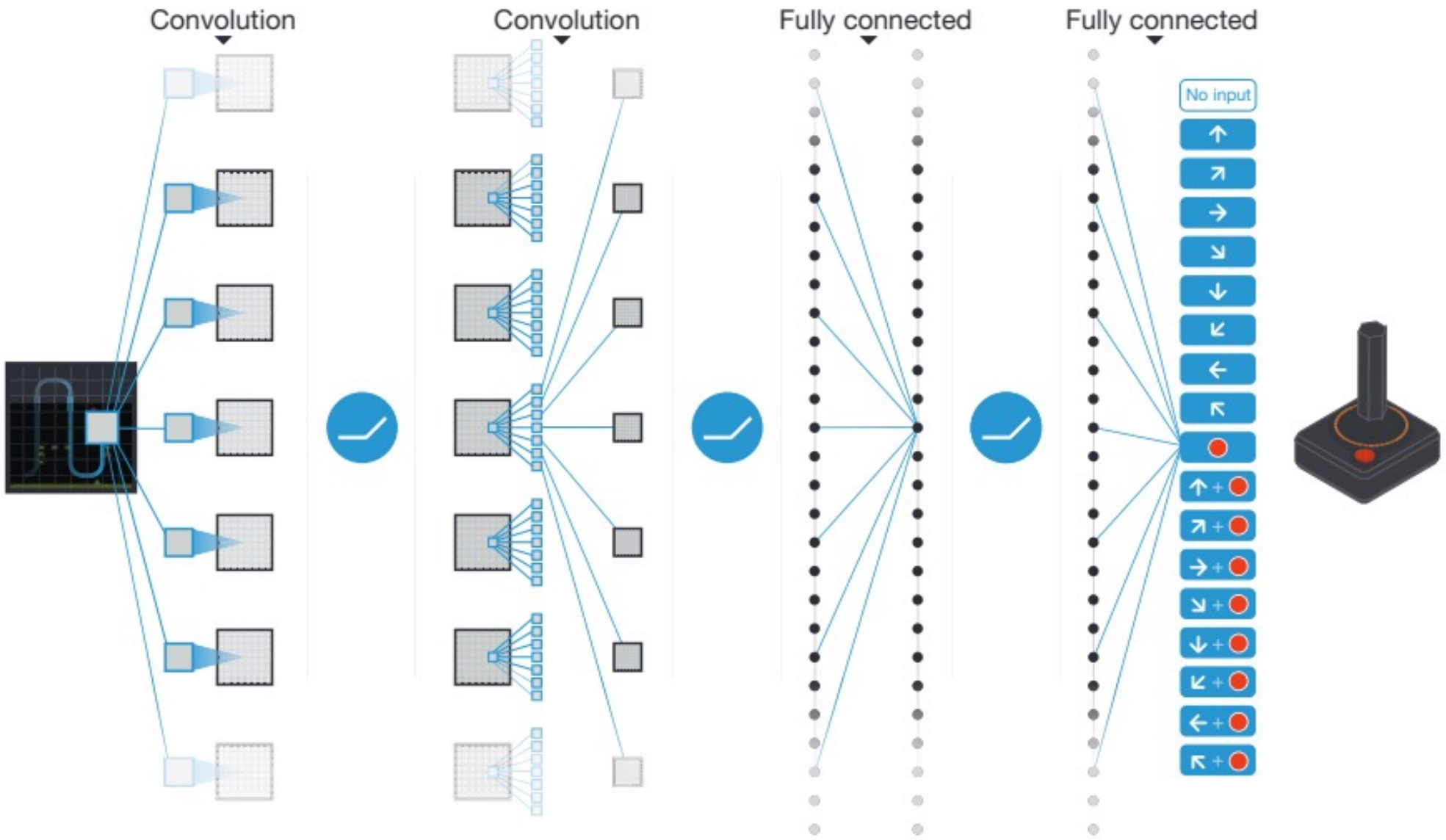
动态环境中为机器人提供实时决策支持

DQN 在 Atari 游戏中的应用

- 端到端学习：直接从游戏像素（状态 s）中学习动作价值函数 Q(s, a)
- 输入状态：由最近 4 帧原始像素图像堆叠而成
- 输出：对应 18 种摇杆/按钮位置的 Q 值（每个动作的预期收益）
- 奖励：每一步的得分变化（即当前帧得分减去前一帧得分）



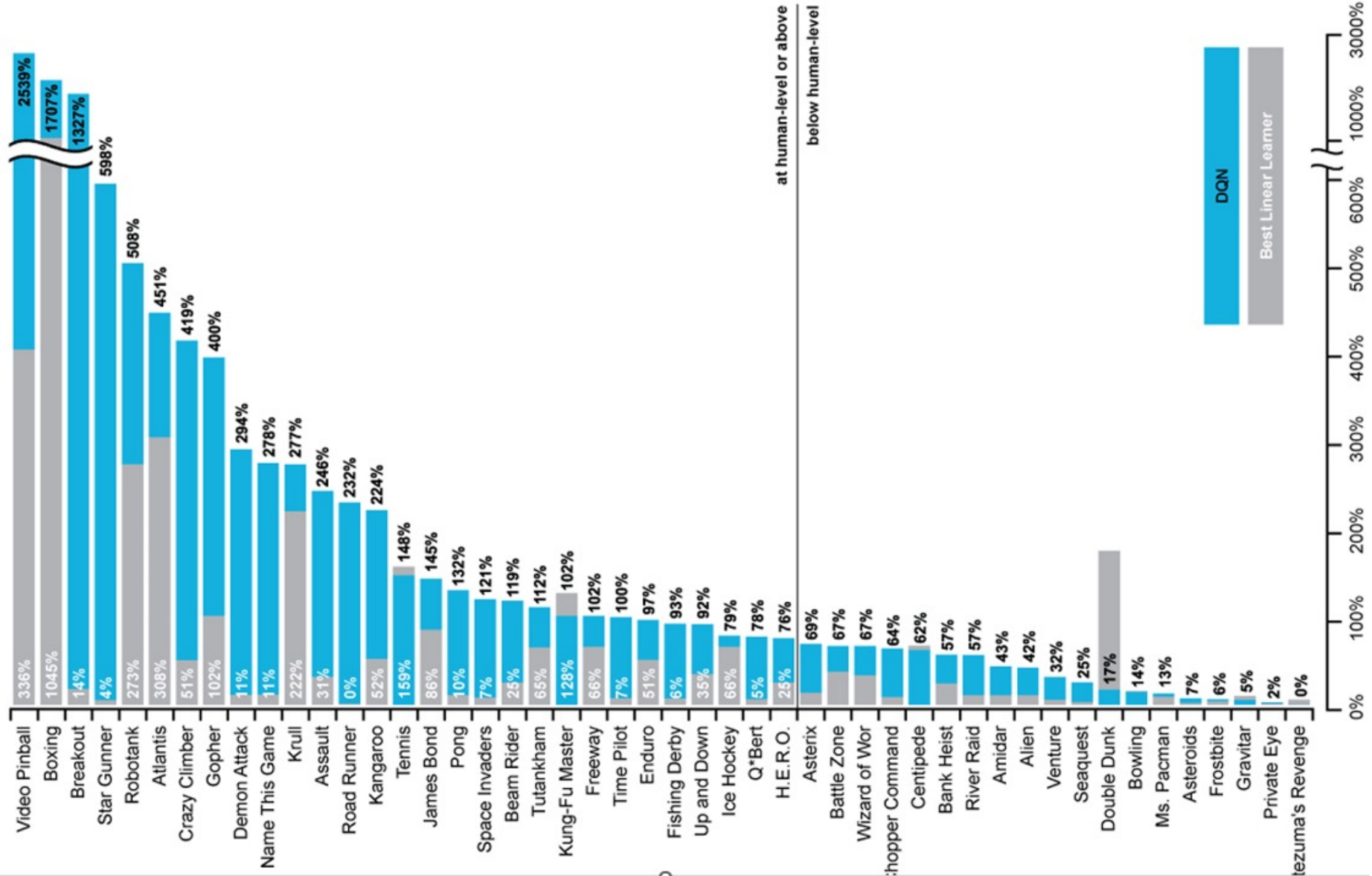
DQN 在 Atari 游戏中的应用



DQN 在 Atari 游戏中的应用

DQN 在 Atari 中表现

- DQN 算法与最佳线性学习器在不同游戏中的表现水平



DQN 表现

- DQN 算法在五款游戏中不同配置的表现：使用经验回放的 Fixed-Q 算法在多数游戏中显著优于无回放的 Q-learning，尤其在 Fixed-Q 中引入回放机制后性能提升幅度最大，经验回放能有效稳定训练并提高数据利用率。

	Replay Fixed-Q	Replay Q-learning	No replay Fixed-Q	No replay Q-learning
Breakout	316.81	240.73	10.16	3.17
Enduro	1006.3	831.25	141.89	29.1
River Raid	7446.62	4102.81	2867.66	1453.02
Seaquest	2894.4	822.55	1003	275.81
Space Invaders	1088.94	826.33	373.22	301.99

Q-learning 结合深度学习所面临的问题

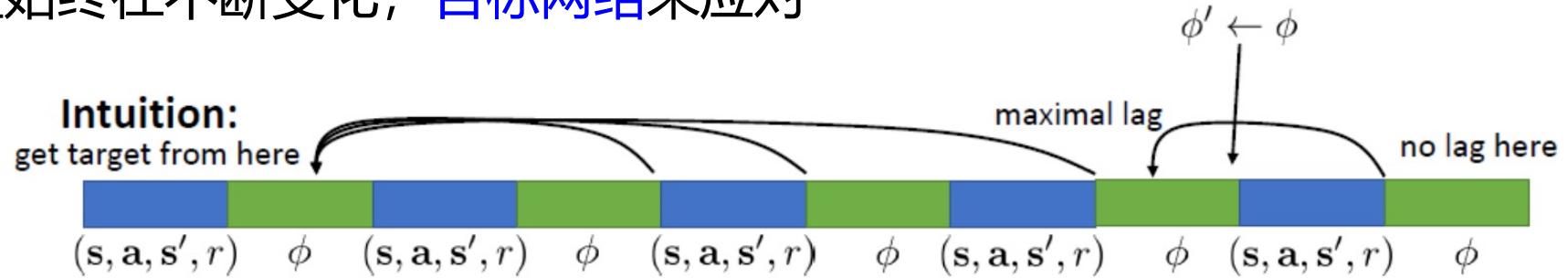
- Q-learning 中直接用神经网络来估计 Q 函数，会面临以下问题：

online Q iteration algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$
2. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)])$

  - 序列状态之间高度相关，经验回放来应对

  - 目标值始终在不断变化，目标网络来应对



目录

DQN 的经验回放

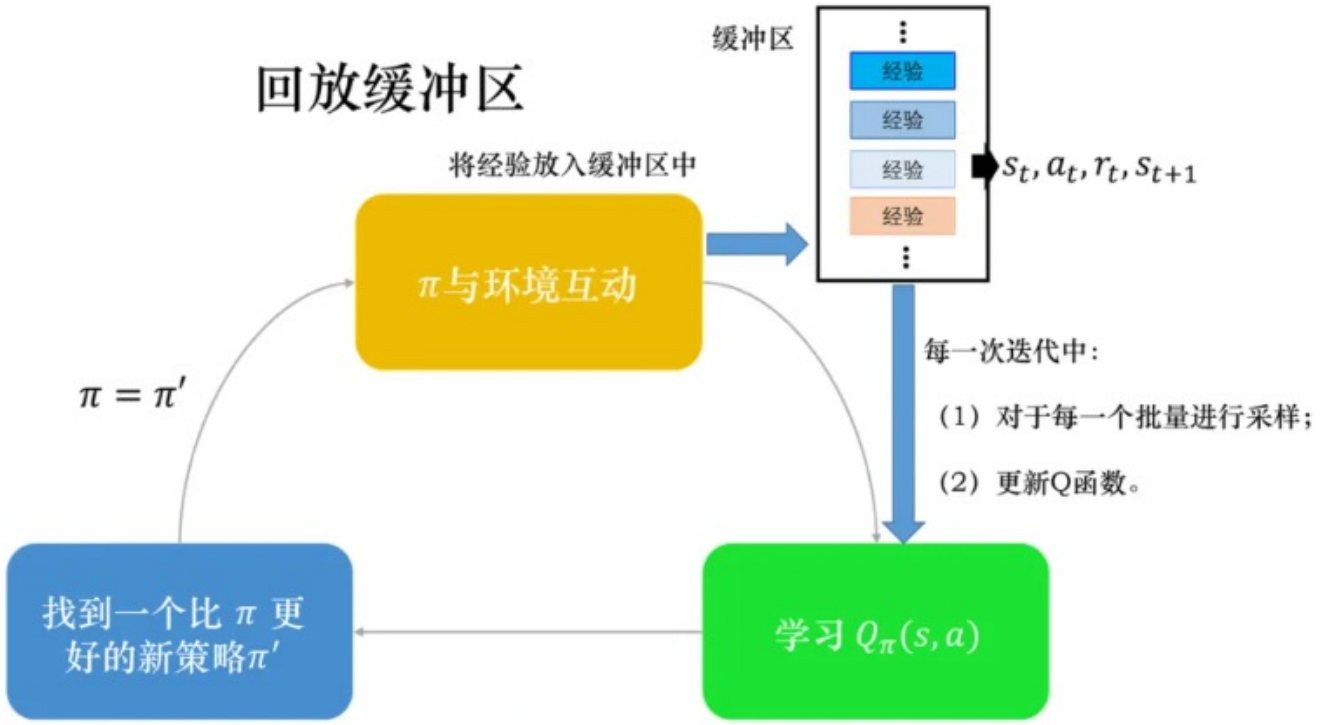
为了提高数据利用效率并减少相关性，DQN引入了经验回放机制。

- **经验回放（Experience Replay）**：旨在改善学习的效率和稳定性。在实时与环境交互中获得的经验（状态、动作、奖励等）通常会被立即用于更新模型。

- **经验回放的作用**：传统的强化学习算法会依赖于时间步之间的相关性，这会导致算法收敛变慢或者不稳定。通过经验回放，我们可以通过随机抽取过去的经验来打破这些相关性，从而提高训练的稳定性。

DQN 工作流程

- **经验存储**：智能体在与环境交互时，将每一步的经验 (s, a, r, s') 存储在一个称为经验池（Replay Buffer）的数据结构中。
- **样本重用**：在每次更新策略时，从经验池中随机采样一小批经验（Mini-Batch），用于更新价值函数或策略。
- **批量更新**：使用随机采样的经验进行批量更新，减少了时间相关性，提高了训练的稳定性。



经验回放的优点

在使用经验回放时，智能体与环境交互产生的状态、动作、奖励和下一状态四元组（s, a, r, s'）被保存在经验回放缓冲区（一个大型数据结构，通常是队列）中。

在模型训练时，从这个缓冲区中随机抽取一批四元组，用它们来更新模型的参数。

优点：

- **数据复用**：同一个经验可以被多次用于训练，提高数据效率。
- **去相关**：随机抽样打破了数据之间的时间相关性，有助于训练稳定。
- **更平滑的学习过程**：缓冲区内的多样性经验可以帮助模型更全面地学习

经验回放的应用

- **Atari 游戏:** DQN 在 Atari 游戏中取得了巨大的成功，经验回放在其中起到了至关重要的作用。智能体通过玩游戏收集经验，存储到回放缓冲区，并从中随机采样进行训练，最终学会了超越人类水平的游戏策略。

- **机器人控制:** 在机器人控制任务中，经验回放可以帮助机器人学习复杂的动作和策略。例如，训练机器人抓取物体，机器人可以通过不断尝试和错误积累经验，并利用经验回放来训练控制策略。

- **自动驾驶:** 在自动驾驶领域，经验回放可以用于训练自动驾驶系统。通过模拟驾驶或真实驾驶收集的经验可以用于训练深度强化学习模型，使其能够学习安全和高效的驾驶策略。

目录

DQN：目标网络

- **目标网络（Target Network）**：使用两个网络：一个用于生成 Q 值（当前网络），另一个用于计算目标 Q 值（目标网络）。目标网络是每隔固定时间更新一次，从而提高稳定性。
  - 在训练过程中，目标网络的参数定期从 Q 网络复制而来，而不是在每次迭代中更新。这有助于稳定训练过程，避免因 Q 网络的频繁更新而导致目标值的剧烈波动。

- 目标：最小化以下损失函数：

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right]$$

- 其中 θ^- 为目标网络的参数，θ 为当前网络的参数。

目标网络工作原理

- **双网络架构**：DQN中存在两个结构相同的神经网络——主网络（MainNet）和目标网络（TargetNet）。
  - 主网络负责实时更新参数并选择动作，而目标网络用于计算 Q 学习的目标值。

- **参数同步机制**：
  - 定期硬更新：每经过固定步数（如1000步），将主网络的参数完全复制到目标网络。
  - 软更新（Polyak平均）：使用滑动平均逐步更新目标网络参数，公式为：

$$\theta^{-} \leftarrow \tau\theta + (1-\tau)\theta^{-}$$

其中τ为更新速率（如0.001），使目标网络参数缓慢逼近主网络

目标网络工作原理

目标值的计算

- 目标网络的输出用于 Q 学习中的时序差分目标（Temporal-Difference Target），即：

$$Q_{\text{target}}(s', a') = r + \gamma \max_{a'} Q_{\text{target}}(s', a'; \theta^-)$$

目标网络参数 θ^- 在一段时间内固定，从而避免主网络频繁更新导致的 Q 值震荡

$$\mathcal{L}_i(w_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}_i}\left[\left(\underbrace{r + \gamma \max_{a'} Q(s', a'; w_i^-)}_{\text{Q-learning target}} - \underbrace{Q(s, a; w_i)}_{\text{Q-network}}\right)^2\right]$$

目标网络的优势

- **训练稳定性**：
  - 目标网络通过延迟参数更新，减少了目标值与当前 Q 值的相关性，防止训练过程中因目标值快速变化导致的震荡。实验表明，在Atari游戏中引入目标网络后，智能体的平均得分提升了 30% 以上。
- **缓解过估计（Overestimation）**：
  - 传统 Q-learning 因最大化操作易高估 Q 值，而目标网络的固定参数可部分抑制这一偏差。例如，在 Double DQN 中结合目标网络，进一步将过估计误差降低了 40%。
- **与经验回放的协同效应**：
  - 目标网络与经验回放机制结合，打破数据的时间相关性，使Q网络能更高效地从历史经验中学习。研究表明，这种组合可将训练效率提升约 25%。

目录

完整 DQN 算法

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

For episode = 1, M **do**

Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

For t = 1,T **do**

With probability ε select a random action a_t

otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

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 D

Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$

Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ

Every C steps reset $\hat{Q} = Q$

End For

End For

DQN 优缺点：

优点：

- 处理高维状态空间
  - DQN 通过深度神经网络直接从高维输入（如图像、传感器数据）学习 Q 值函数，突破了传统 Q-learning 在状态空间过大时的“维度灾难”限制。
- 训练稳定性提升（经验回放与目标网络）
  - 经验回放：通过存储和随机采样历史交互数据，打破数据时序相关性，提高样本利用率，减少训练震荡。
  - 目标网络：使用独立网络计算目标Q值，避免主网络参数频繁更新导致目标值波动，显著提升收敛性

缺点：

- 仅支持离散动作空间
  - DQN 通过最大化 Q 值选择动作，天然适用于离散动作（如“左转”或“跳跃”），但无法直接处理连续动作。
- 样本效率低
  - 依赖大量交互数据，尤其在复杂任务（如机器人导航）中需数百万步训练，物理世界应用成本高昂。例如，自动驾驶中需反复模拟或实车测试

DQN 和 Q-learning 联系

- **核心思想相同**：两者都基于 **Q-learning 算法框架**，目标是学习一个最优动作价值函数 $Q(s,a)$，通过最大化长期累积奖励选择动作。

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

α 是学习率，γ 是折扣因子

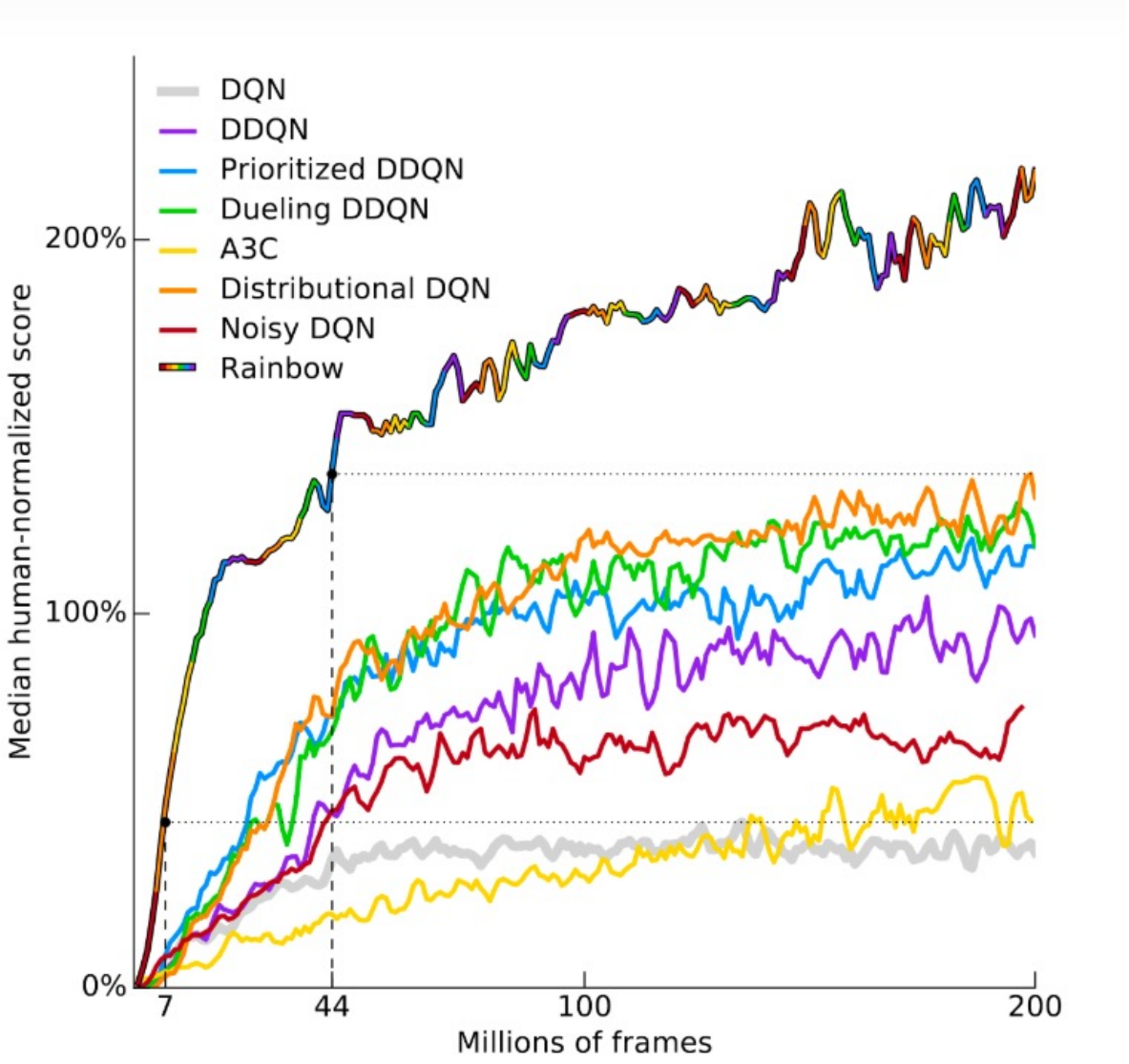
DQN 继承此公式，但用神经网络代替表格计算 $Q(s,a)$，并采用经验回复和目标网络来应对 Q-learning 中直接引入神经网络所带来的问题

DQN 和 Q-learing 对比

特性	Q-learning	DQN
状态空间	低维离散（如坐标）	高维连续（如图像、传感器数据）
Q值存储方式	表格	神经网络
数据使用	按顺序更新当前经验	经验回放 + 随机采样
目标值计算	当前网络参数	独立目标网络（参数延迟更新）
适用场景	简单环境（迷宫、棋类）	复杂环境（游戏、机器人控制）
训练稳定性	低（易发散）	高（经验回放 + 目标网络）

DQN 的扩展：彩虹（rainbow）

- **Double DQN**：解决了 Q 值过高估计的问题，提高了学习的稳定性。
- **Prioritized Experience Replay**：提高了数据利用率，加快了学习速度。
- **Dueling Network Architecture**：提高了泛化能力，使得智能体更容易学习到不同动作的价值。
- **Multi-step Learning**：平衡了偏差和方差，加速了奖励的传播。
- **Distributional Q-learning**：提供了更丰富的信息，提高了学习的鲁棒性。
- **Noisy DQN**：提高了探索效率，尤其是在奖励稀疏的环境中



课后作业

思考题：

1. 经验回放（Experience Replay）的作用是什么？为什么需要打破数据相关性？
2. DQN 中的目标网络（Target Network）如何设计？其必要性是什么？

补充：从 DQN 到策略式方法（Policy-Based Methods）

DQN 局限：

- 只能输出离散动作（对 DQN 的改进 DDPG 可以输出连续动作）
- 估计误差随动作空间维度爆炸
- 只是训练确定性/类确定性策略，难以应对大规模问题里的状态重名问题

策略式强化学习思路：直接优化参数化策略

- 令策略 $\pi_\theta(a \mid s)$ 可微
- 以期望回报 $J(\theta) = E_{\pi_\theta}[\sum_{t=0}^{\infty}\gamma^t r_t]$ 为目标
- 求 $\nabla_\theta J(\theta)$，对应为策略梯度（Policy Gradient）
- 再梯度上升计算 $\theta^* = argmax_\theta J(\theta)$，从而得到最优的策略 π_{θ^*}

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

补充：策略梯度定理（Policy Gradient Theorem）

无基线版

$$\nabla_\theta J(\theta) = \sum_s d_{\pi_\theta}(s) \sum_a Q^{\pi_\theta}(s,a)\, \nabla_\theta \pi_\theta(a \mid s) = \mathbb{E}_{S_t \sim d_{\pi_\theta}, A_t \sim \pi_\theta}\big[\, Q^{\pi_\theta}(S_t, A_t)\, \nabla_\theta \ln \pi_\theta(A_t \mid S_t)\big]$$

$d_{\pi_\theta}(s)$：在策略 π_θ 下对状态 s 的稳态分布

$$J(\theta) = \sum_s d_{\pi_\theta}(s) \sum_a Q^{\pi_\theta}(s,a) \pi_\theta(a \mid s)$$

有基线版

$$\nabla_\theta J(\theta) = \mathbb{E}_{S_t \sim d_{\pi_\theta}, A_t \sim \pi_\theta}\left[(Q^{\pi_\theta}(S_t, A_t) - \underbrace{b(S_t)}_{\text{baseline}})\, \nabla_\theta \ln \pi_\theta(A_t \mid S_t) \right]$$

补充：策略梯度方法一览

$$\nabla_\theta J(\theta) = \mathbb{E}_{S_t\sim d_{\pi_\theta}, A_t\sim\pi_\theta}\left[(Q^{\pi_\theta}(S_t,A_t) - \underbrace{b(S_t)}_{\text{baseline}})\nabla_\theta \ln \pi_\theta(A_t \mid S_t)\right]$$

$$\begin{aligned} E_{\pi_\theta}[\delta^{\pi_\theta} \mid s,a] &= E_{\pi_\theta}[r+\gamma V^{\pi_\theta}(s') \mid s,a] - V^{\pi_\theta}(s) \\ &= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) \\ &= A^{\pi_\theta}(s,a) \end{aligned}$$

算法	采样与目标	取 baseline $b(S_t)$	近似 Q^π / A^π	更新公式（每步增量形式）
REINFORCE	完整轨迹回报 G_t	0	$Q^\pi \approx G_t$	$\theta \leftarrow \theta + \alpha G_t \nabla_\theta \ln \pi$
REINFORCE + Baseline	同上	任意 b(s)（常取 V^π）	$Q^\pi - b(s)$	$\theta \leftarrow \theta + \alpha(G_t - b(s_t))\nabla_\theta \ln \pi$
Actor–Critic (AC)	在线 TD 误差 δ_t	V_w（由 Critic 学习）	$\delta_t = r_t + \gamma V_w(s') - V_w(s)$	$\begin{cases} w \leftarrow w - \alpha_v \delta_t \nabla_w V_w \\ \theta \leftarrow \theta + \alpha_a \delta_t \nabla_\theta \ln \pi \end{cases}$
A2C / A3C	多环境 (sync/async) GAE 优势 $\hat{A}_t$	V_w	$\hat{A}_t$ (GAE, λ-return)	同上，批量或线程并行
PPO	同 A2C, 再加剪切比率	V_w	$\hat{A}_t$	$w \leftarrow w - \alpha_v \widehat{\nabla}_w \mathbb{E}[(V_w - \hat{G}_t)^2]$ $\theta \leftarrow \theta + \alpha_\theta \widehat{\nabla}_\theta \mathcal{L}$

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \qquad L^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)\,\hat{A}_t,\ \text{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)\,\hat{A}_t\big)\Big] \qquad \mathcal{L}(\theta, w) = L^{\text{CLIP}}(\theta) \ - \ c_v\,\mathbb{E}_t\big[(V_w(s_t) - \hat{G}_t)^2\big] \ + \ c_e\,\mathbb{E}_t\big[\mathcal{H}\big(\pi_\theta(\cdot \mid s_t)\big)\big]$$

谢谢！