Kai Arulkumaran, Marc Peter Deisenroth,
Miles Brundage, and Anil Anthony Bharath

Deep Reinforcement Learning

A brief survey

Deep reinforcement learning (DRL) is poised to revolutionize the field of artificial intelligence (AI) and represents a step toward building autonomous systems with a higher-level understanding of the visual world. Currently, deep learning is enabling reinforcement learning (RL) to scale to problems that were previously intractable, such as learning to play video games directly from pixels. DRL algorithms are also applied to robotics, allowing control policies for robots to be learned directly from camera inputs in the real world. In this survey, we begin with an introduction to the general field of RL, then progress to the main streams of value-based and policy-based methods. Our survey will cover central algorithms in deep RL, including the deep Q-network (DQN), trust region policy optimization (TRPO), and asynchronous advantage actor critic. In parallel, we highlight the unique advantages of deep neural networks, focusing on visual understanding via RL. To conclude, we describe several current areas of research within the field.

Introduction

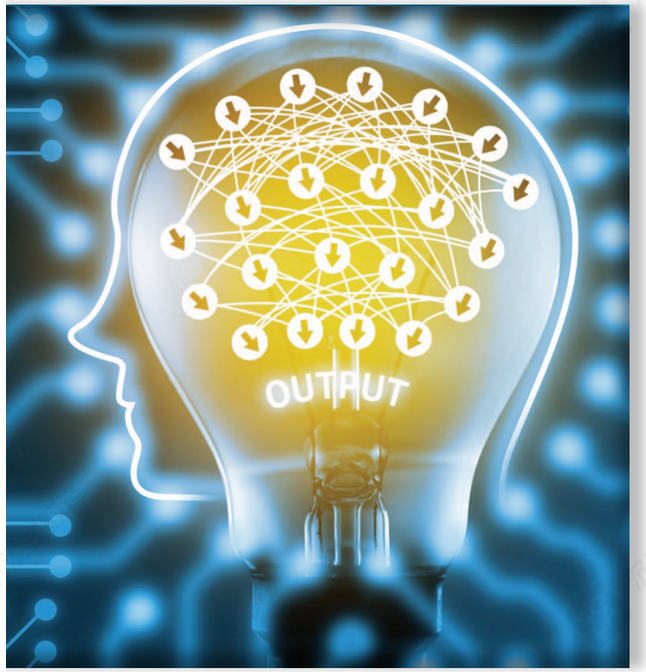
One of the primary goals of the field of AI is to produce fully autonomous agents that interact with their environments to learn optimal behaviors, improving over time through trial and error. Crafting AI systems that are responsive and can effectively learn has been a long-standing challenge, ranging from robots, which can sense and react to the world around them, to purely software-based agents, which can interact with natural language and multimedia. A principled mathematical framework for experience-driven autonomous learning is RL [78]. Although RL had some successes in the past [31], [53], [74], [81], previous approaches lacked scalability and were inherently limited to fairly low-dimensional problems. These limitations exist because RL algorithms share the same complexity issues as other algorithms: memory complexity, computational complexity, and, in the case of machine-learning algorithms, sample complexity [76]. What we have witnessed in recent years—the rise of deep learning, relying on the powerful function approximation and representation learning properties of deep neural networks—has provided us with new tools to overcoming these problems.

Digital Object Identifier 10.1109/MSP.2017.2743240
Date of publication: 13 November 2017

Kai Arulkumaran、Marc Peter Deisenroth、
Miles Brundage 和 Anil Anthony Bharath

深度强化学习

简短综述



Deep reinforcement learning (DRL) is poised to revolutionize 人工智能 (AI) 的领域，并代表着向构建具有更高层次视觉世界理解能力的自主系统迈进的一步。目前，深度学习使强化学习 (RL) 能够扩展到以前难以解决的问题，例如直接从像素中学习玩电子游戏。DRL 算法也应用于机器人技术，允许机器人的控制策略直接从现实世界中的摄像头输入中学习。在本综述中，我们从强化学习领域的总体介绍开始，然后进展到基于价值和基于策略的主要方法。我们的综述将涵盖深度强化学习中的核心算法，包括深度 Q- 网络 (DQN) 、信任域策略优化 (TRPO) 和异步优势演员评论家。同时，我们强调了深度神经网络的优势，重点关注通过强化学习的视觉理解。最后，我们描述了该领域内几个当前的研究方向。

引言

人工智能领域的一个主要目标是为环境中的完全自主代理提供支持，使其通过与环境互动来学习最佳行为，并通过试错不断改进。设计和制造能够响应并有效学习的 AI 系统一直是一个长期存在的挑战，从能够感知和反应周围世界的机器人，到能够与自然语言和多媒体互动的纯软件代理。一种基于经验的自主学习原理数学框架是强化学习（ RL [78] ）。尽管强化学习在过去取得了一些成功 [31], [53], [74], [81], ，但之前的强化学习方法缺乏可扩展性，并且本质上仅限于低维问题。这些限制的存在是因为强化学习算法与其他算法一样，存在相同的复杂性问题：内存复杂性、计算复杂性和，对于机器学习算法而言，样本复杂性 [76]。近年来我们所见证的 —— 深度学习的兴起，它依赖于深度神经网络强大的函数逼近和表示学习特性 —— 为我们提供了新的工具来克服这些问题。

数字对象标识符 *10.1109/MSP.2017.2743240* 出版日期：*2017* 年 *11* 月 *13* 日

The advent of deep learning has had a significant impact on many areas in machine learning, dramatically improving the state of the art in tasks such as object detection, speech recognition, and language translation [39]. The most important property of deep learning is that deep neural networks can automatically find compact low-dimensional representations (features) of high-dimensional data (e.g., images, text, and audio). Through crafting inductive biases into neural network architectures, particularly that of hierarchical representations, machine-learning practitioners have made effective progress in addressing the curse of dimensionality [7]. Deep learning has similarly accelerated progress in RL, with the use of deep-learning algorithms within RL defining the field of DRL. The aim of this survey is to cover both seminal and recent developments in DRL, conveying the innovative ways in which neural networks can be used to bring us closer toward developing autonomous agents. For a more comprehensive survey of recent efforts in DRL, we refer readers to the overview by Li [43].

Deep learning enables RL to scale to decision-making problems that were previously intractable, i.e., settings with high-dimensional state and action spaces. Among recent work in the field of DRL, there have been two outstanding success stories. The first, kickstarting the revolution in DRL, was the development of an algorithm that could learn to play a range of Atari 2600 video games at a superhuman level, directly from image pixels [47]. Providing solutions for the instability of function approximation techniques in RL, this work was the first to convincingly demonstrate that RL agents could be trained on raw, high-dimensional observations, solely based on a reward signal. The second standout success was the development of a hybrid DRL system, AlphaGo, that defeated a human world champion in Go [73], paralleling the historic achievement of IBM's Deep Blue in chess two decades earlier [9]. Unlike the handcrafted rules that have dominated chess-playing systems, AlphaGo comprised neural networks that were trained using supervised learning and RL, in combination with a traditional heuristic search algorithm.

DRL algorithms have already been applied to a wide range of problems, such as robotics, where control policies for robots can now be learned directly from camera inputs in the real world [41], [42], succeeding controllers that used to be hand-engineered or learned from low-dimensional features of the robot's state. In Figure 1, we showcase just some of the domains that DRL has

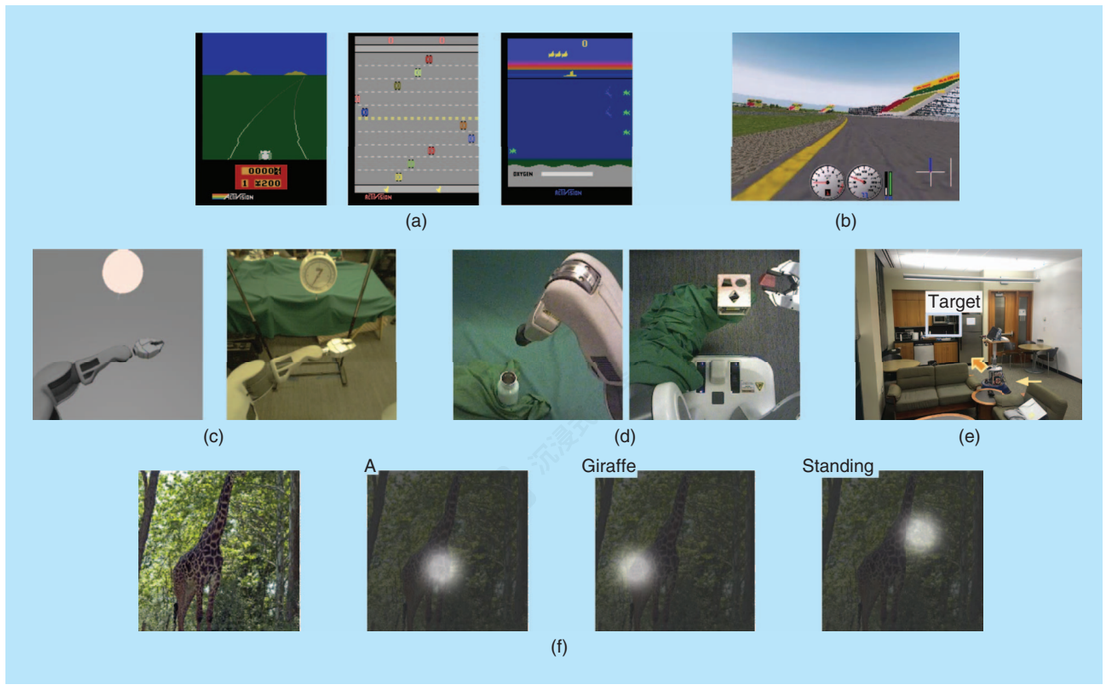


FIGURE 1. A range of visual RL domains. (a) Three classic Atari 2600 video games, *Enduro, Freeway,* and *Seaquest,* from the Arcade Learning Environment (ALE) [5]. Due to the range of supported games that vary in genre, visuals, and difficulty, the ALE has become a standard test bed for DRL algorithms [20], [47], [48], [55], [70], [75], [92]. The ALE is one of several benchmarks that are now being used to standardize evaluation in RL. (b) The TORCS car racing simulator, which has been used to test DRL algorithms that can output continuous actions [33], [44], [48] (as the games from the ALE only support discrete actions). (c) Utilizing the potentially unlimited amount of training data that can be amassed in robotic simulators, several methods aim to transfer knowledge from the simulator to the real world [11], [64], [84]. (d) Two of the four robotic tasks designed by Levine et al. [41]: screwing on a bottle cap and placing a shaped block in the correct hole. Levine et al. [41] were able to train visuomotor policies in an end-to-end fashion, showing that visual servoing could be learned directly from raw camera inputs by using deep neural networks. (e) A real room, in which a wheeled robot trained to navigate the building is given a visual cue as input and must find the corresponding location [100]. (f) A natural image being captioned by a neural network that uses RL to choose where to look [99]. (b)–(f) reproduced from [41], [44], [84], [99], and [100], respectively.

深度学习的出现对机器学习的许多领域产生了重大影响，在目标检测、语音识别和语言翻译等任务中显著提升了当前技术水平 [39]。深度学习最重要的特性是深度神经网络可以自动找到高维数据（例如图像、文本和音频）的紧凑低维表示（特征）。通过将归纳偏差构建到神经网络架构中，特别是分层表示中，机器学习从业者有效地解决了维度灾难问题 [7]。深度学习同样加速了强化学习（ RL ）的进展，其中在 RL 中使用深度学习算法定义了深度强化学习（ DRL ）领域。本调查旨在涵盖 DRL 的里程碑式和近期发展，传达神经网络如何创新地用于推动我们更接近开发自主智能体。对于更全面的 DRL 近期工作综述，我们建议读者参考 Li 的概述 [43]。

深度学习使强化学习能够扩展到以前难以处理的决策问题，即具有高维状态和动作空间的环境。在最近的研究工作中

在深度强化学习领域，有两个杰出的成功案例。第一个，开启了深度强化学习革命的，是开发了一种能够从图像像素 [47] 直接学习玩一系列 Atari 2600 视频游戏并以超人类水平进行的算法。这项工作为强化学习中函数逼近技术的不稳定性提供了解决方案，并且是第一个有说服力地证明强化学习智能体可以在原始、高维观察上进行训练，仅基于奖励信号的工作。第二个杰出的成功是开发了一个混合深度强化学习系统 AlphaGo，它在围棋比赛中击败了人类世界冠军 [73], ，与二十年前 IBM 的深蓝在象棋比赛中取得的历史性成就相呼应 [9]。与主导象棋系统的手工规则不同，AlphaGo 包含了使用监督学习和强化学习进行训练的神经网络，并结合了传统的启发式搜索算法。

深度强化学习算法已经应用于广泛的问题，例如机器人学，现在机器人的控制策略可以直接从现实世界的摄像头输入中学习 [41], [42], ，而以前这些控制器要么是手工设计的，要么是从机器人状态的低维特征中学习的。在图 1 中，我们展示了深度强化学习已经涉足的一些领域

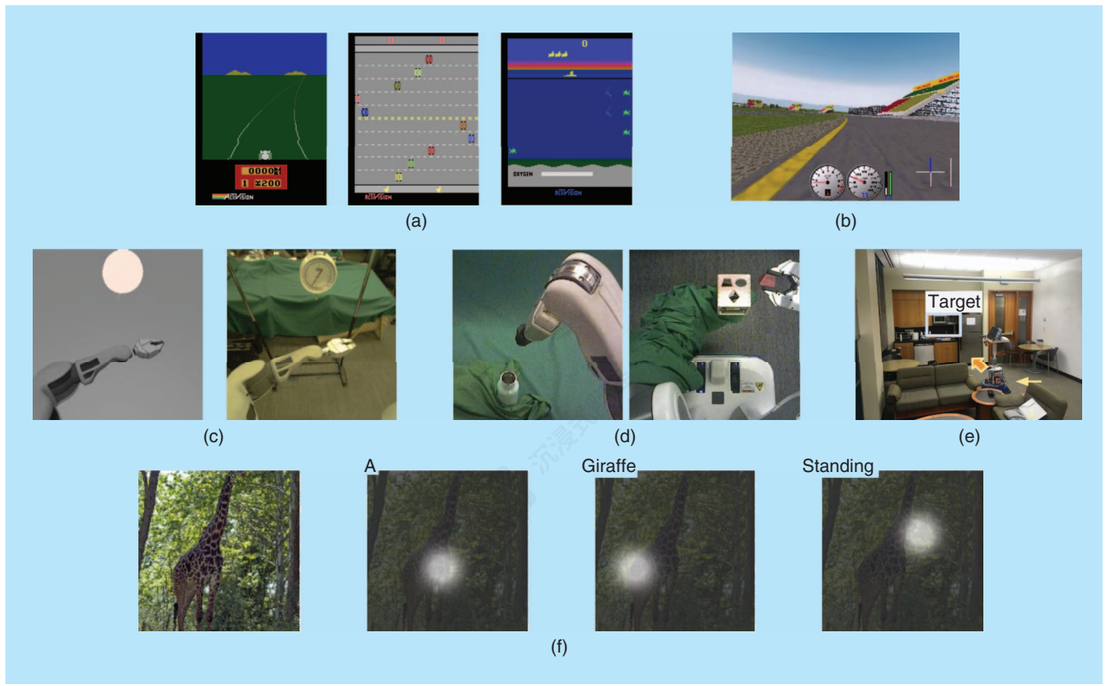


图 1。 一系列视觉强化学习领域。(a) 来自街机学习环境 (ALE)[5] 的三个经典雅达利 2600 视频游戏，*Enduro* 、 *Freeway*，和 *Seaquest*，。由于支持的游戏种类、视觉效果和难度各不相同，ALE 已经成为深度强化学习算法的标准测试平台 [20], [47], [48], [55], [70], [75], [92]。ALE 是现在用于标准化强化学习评估的几个基准之一。(b) TORCS 赛车模拟器，它被用于测试可以输出连续动作的深度强化学习算法 [33], [44], [48] （因为 ALE 中的游戏只支持离散动作）。(c) 利用机器人模拟器中可以积累的潜在无限训练数据，几种方法旨在将知识从模拟器转移到现实世界 [11], [64], [84]。(d) Levine 等人 [41] 设计的四个机器人任务中的两个：拧瓶盖和将形状块放在正确的孔中。Levine 等人 [41] 能够以端到端的方式训练视觉运动策略，表明通过使用深度神经网络，视觉伺服可以直接从原始摄像头输入中学习。(e) 一个真实房间，一个轮式机器人被训练在建筑物中导航，它被给予视觉提示作为输入，必须找到相应的位置 [100]。(f) 一个自然图像被一个使用强化学习来选择注视位置的神经网络所描述 [99]。(b)–(f) 分别重印自 [41], [44], [84], [99], 和 [100], 。

been applied to, ranging from playing video games [47] to indoor navigation [100].

Reward-driven behavior

Before examining the contributions of deep neural networks to RL, we will introduce the field of RL in general. The essence of RL is learning through interaction. An RL agent interacts with its environment and, upon observing the consequences of its actions, can learn to alter its own behavior in response to rewards received. This paradigm of trial-and-error learning has its roots in behaviorist psychology and is one of the main foundations of RL [78]. The other key influence on RL is optimal control, which has lent the mathematical formalisms (most notably dynamic programming [6]) that underpin the field.

In the RL setup, an autonomous agent, controlled by a machine-learning algorithm, observes a state $\mathbf{s}_t$ from its environment at time step t. The agent interacts with the environment by taking an action $\mathbf{a}_t$ in state $\mathbf{s}_t$. When the agent takes an action, the environment and the agent transition to a new state, $\mathbf{s}_{t+1}$, based on the current state and the chosen action. The state is a sufficient statistic of the environment and thereby comprises all the necessary information for the agent to take the best action, which can include parts of the agent such as the position of its actuators and sensors. In the optimal control literature, states and actions are often denoted by $\mathbf{x}_t$ and $\mathbf{u}_t$, respectively.

The best sequence of actions is determined by the rewards provided by the environment. Every time the environment transitions to a new state, it also provides a scalar reward r_{t+1} to the agent as feedback. The goal of the agent is to learn a policy (control strategy) π that maximizes the expected return (cumulative, discounted reward). Given a state, a policy returns an action to perform; an optimal policy is any policy that maximizes the expected return in the environment. In this respect, RL aims to solve the same problem as optimal control. However, the challenge in RL is that the agent needs to learn about the consequences of actions in the environment by trial and error, as, unlike in optimal control, a model of the state transition dynamics is not available to the agent. Every interaction with the environment yields information, which the agent uses to update its knowledge. This perception-action-learning loop is illustrated in Figure 2.

Markov decision processes

Formally, RL can be described as a Markov decision process (MDP), which consists of

- a set of states $\mathcal{S}$, plus a distribution of starting states $p(\mathbf{s}_0)$
- a set of actions $\mathcal{A}$
- transition dynamics $\mathcal{T}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ that map a state-action pair at time t onto a distribution of states at time $t+1$
- an immediate/instantaneous reward function $\mathcal{R}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$
- a discount factor $\gamma \in [0, 1]$, where lower values place more emphasis on immediate rewards.

In general, the policy π is a mapping from states to a probability distribution over actions $\pi: \mathcal{S} \rightarrow p(\mathcal{A} = \mathbf{a} | \mathcal{S})$. If the MDP is episodic, i.e., the state is reset after each episode of length T, then the sequence of states, actions, and rewards in an episode constitutes a trajectory or rollout of the policy. Every rollout of a policy accumulates rewards from the environment, resulting in the return $R = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$. The goal of RL is to find an optimal policy, π^* that achieves the maximum expected return from all states:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}[R | \pi]. \tag{1}$$

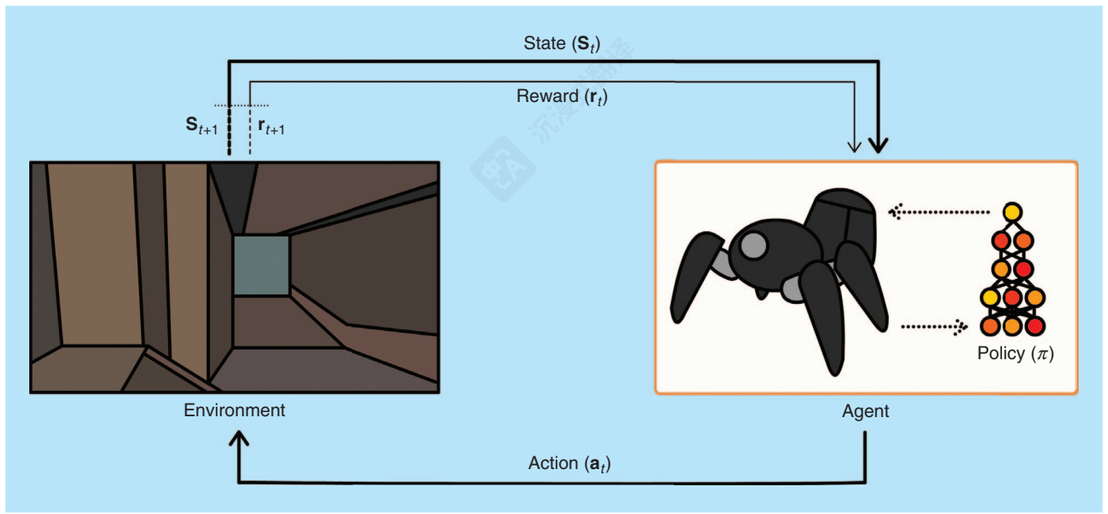


FIGURE 2. The perception-action-learning loop. At time t, the agent receives state $\mathbf{s}_t$ from the environment. The agent uses its policy to choose an action $\mathbf{a}_t$. Once the action is executed, the environment transitions a step, providing the next state, $\mathbf{s}_{t+1}$, as well as feedback in the form of a reward, r_{t+1}. The agent uses knowledge of state transitions, of the form $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_{t+1})$, to learn and improve its policy.

已被应用于，范围从玩电子游戏 [47] 到室内导航 [100]。

奖励驱动行为

在考察深度神经网络对强化学习（RL）的贡献之前，我们将介绍强化学习领域的一般情况。强化学习的本质是通过交互进行学习。强化学习智能体与其环境交互，并在观察到其行为的结果后，可以学习改变其行为以响应所获得的奖励。这种试错学习范式源于行为主义心理学，是强化学习的主要基础之一 [78]。对强化学习的另一个关键影响是最优控制，它提供了支撑该领域的数学形式化（最著名的是动态规划 [6]）。

在强化学习（RL）设置中，一个自主代理，由机器学习算法控制，观察其环境在时间步长 $\mathbf{s}_t$ 处的状态。t 代理通过与环境在状态 $\mathbf{a}_t$ 处采取行动。$_t\mathbf{s}$ 进行交互。当代理采取行动时，环境和代理根据当前状态和所选行动过渡到新状态，$\mathbf{s}_{t+1}$。状态是环境的充分统计量，因此包含代理采取最佳行动所需的所有必要信息，这可能包括代理的部分内容，例如其执行器和传感器的位置。在最优控制文献中，状态和行动通常分别表示为 $\mathbf{x}_t$ 和，$\mathbf{u}_t$。

最佳行动序列由环境提供的奖励决定。每次环境过渡到新状态时，它还会向代理提供标量奖励 r_{t+1} 作为反馈。代理的目标是学习一个策略（控制策略）π，以最大化预期回报（累积、折扣奖励）。给定一个状态，策略会返回一个行动

为了执行；任何能够最大化环境中预期收益的策略都是最优策略。在这方面，强化学习（RL）旨在解决与最优控制相同的问题。然而，强化学习的挑战在于，智能体需要通过试错来学习环境中行动的后果，因为在最优控制中，智能体没有可用的状态转换动态模型。与环境的每一次交互都会产生信息，智能体利用这些信息来更新其知识。这种感知 - 行动 - 学习循环如图 2 所示。

Markov decision processes

形式上，强化学习可以被描述为一个马尔可夫决策过程（MDP），它由以下部分组成

■ 一组状态 ， S plus a starting states distribution ( )p s0■ 一组动作 A■ 转移动态 (,)s s$_{t}$a$_{t}$T t t that map a state-action pair at time t onto a distribution of states at time$_{t}$ t 1■ 即时 / 瞬时奖励函数 (, ,)s asR t$_{t}$t$_{t}$■ 折扣因子 !c where lower values place more emphasis on immediate rewards.

通常情况下，策略 π 是从状态到动作概率分布（| ）的映射。: p $\mathbf{a}\mathcal{S}$ A S"π 如果马尔可夫决策过程是阶段性的，即每个长度为 T 的阶段后状态被重置，那么一个阶段中的状态、动作和奖励序列构成策略的轨迹或展开。策略的每次展开都会从环境中累积奖励，从而产生回报 R r $_{t}$T $_{t}$ 10γ = = + $\sum$ The goal of RL is to find an optimal 策略，$^{*}\pi$，它从所有状态中获得最大预期回报：

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}[R | \pi]. \tag{1}$$

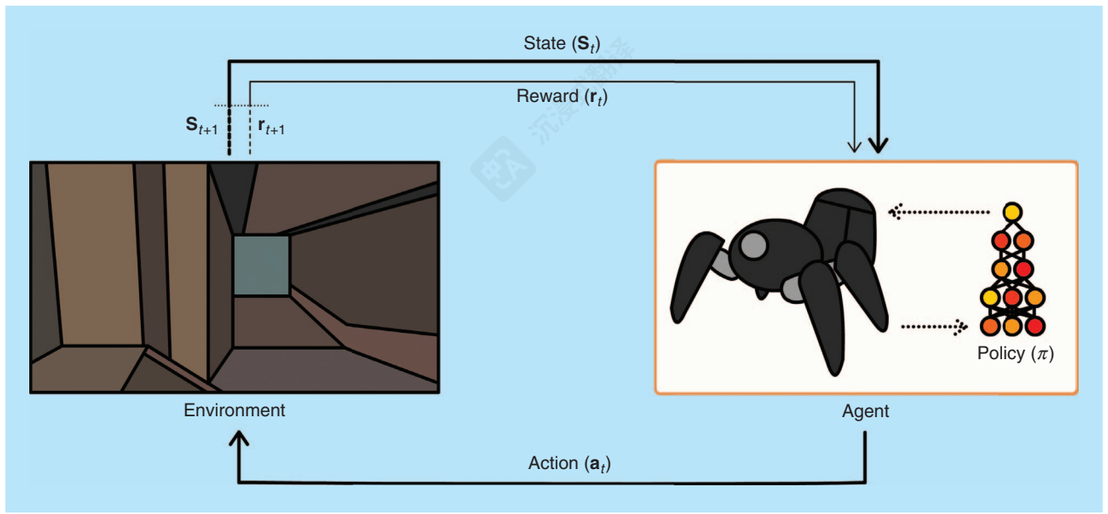


图 2。 感知 - 行动 - 学习循环。在时间 t，智能体从环境中接收状态 $\mathbf{s}_t$。智能体使用其策略选择一个动作。$\mathbf{a}$ 一旦动作被执行，环境转换一步，提供下一个状态，$_{t+}\mathbf{s}$ 以及以奖励形式提供的反馈，$_{t+}r$ 智能体使用状态转换的知识，形式为 $(\ _{,t}\ ,\ _{t+})/\mathbf{s}_{+}$as 来学习和改进其策略。

It is also possible to consider nonepisodic MDPs, where $T = \infty$. In this situation, $\gamma < 1$ prevents an infinite sum of rewards from being accumulated. Furthermore, methods that rely on complete trajectories are no longer applicable, but those that use a finite set of transitions still are.

A key concept underlying RL is the Markov property—only the current state affects the next state, or, in other words, the future is conditionally independent of the past given the present state. This means that any decisions made at $\mathbf{s}_t$ can be based solely on $\mathbf{s}_{t+1}$, rather than $\{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{t-1}\}$. Although this assumption is held by the majority of RL algorithms, it is somewhat unrealistic, as it requires the states to be fully observable. A generalization of MDPs are partially observable MDPs (POMDPs), in which the agent receives an observation $\mathbf{o}_t \in \Omega$, where the distribution of the observation $p(\mathbf{o}_{t+1} \mid \mathbf{s}_{t+1}, \mathbf{a}_t)$ is dependent on the current state and the previous action [27]. In a control and signal processing context, the observation would be described by a measurement/observation mapping in a state-space model that depends on the current state and the previously applied action.

POMDP algorithms typically maintain a belief over the current state given the previous belief state, the action taken, and the current observation. A more common approach in deep learning is to utilize recurrent neural networks (RNNs) [20], [21], [48], [96], which, unlike feedforward neural networks, are dynamical systems.

Challenges in RL

It is instructive to emphasize some challenges faced in RL:

- The optimal policy must be inferred by trial-and-error interaction with the environment. The only learning signal the agent receives is the reward.
- The observations of the agent depend on its actions and can contain strong temporal correlations.
- Agents must deal with long-range time dependencies: often the consequences of an action only materialize after many transitions of the environment. This is known as the (temporal) *credit assignment problem* [78].

We will illustrate these challenges in the context of an indoor robotic visual navigation task: if the goal location is specified, we may be able to estimate the distance remaining (and use it as a reward signal), but it is unlikely that we will know exactly what series of actions the robot needs to take to reach the goal. As the robot must choose where to go as it navigates the building, its decisions influence which rooms it sees and, hence, the statistics of the visual sequence captured. Finally, after navigating several junctions, the robot may find itself in a dead end. There is a range of problems, from learning the consequences of actions to balancing exploration versus exploitation, but ultimately these can all be addressed formally within the framework of RL.

RL algorithms

So far, we have introduced the key formalism used in RL, the MDP, and briefly noted some challenges in RL. In the following, we will distinguish between different classes of RL algorithms. There are two main approaches to solving RL problems: methods based on value functions and methods based on policy search. There is also a hybrid actor-critic approach that employs both value functions and policy search. Next, we will explain these approaches and other useful concepts for solving RL problems.

Value functions

Value function methods are based on estimating the value (expected return) of being in a given state. The state-value function $V^\pi(\mathbf{s})$ is the expected return when starting in state $\mathbf{s}$ and following π subsequently:

$$V^\pi(\mathbf{s}) = \mathbb{E}[R \mid \mathbf{s}, \pi]. \tag{2}$$

The optimal policy, π^*, has a corresponding state-value function $V^*(\mathbf{s})$, and vice versa; the optimal state-value function can be defined as

$$V^*(\mathbf{s}) = \max_\pi V^\pi(\mathbf{s}) \quad \forall \mathbf{s} \in \mathcal{S}. \tag{3}$$

If we had $V^*(\mathbf{s})$ available, the optimal policy could be retrieved by choosing among all actions available at $\mathbf{s}_t$ and picking the action $\mathbf{a}$ that maximizes $\mathbb{E}_{\mathbf{s}_{t+1} \sim \mathcal{T}(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a})}[V^*(\mathbf{s}_{t+1})]$.

In the RL setting, the transition dynamics $\mathcal{T}$ are unavailable. Therefore, we construct another function, the state-action value or quality function $Q^\pi(\mathbf{s}, \mathbf{a})$, which is similar to V^π, except that the initial action $\mathbf{a}$ is provided and π is only followed from the succeeding state onward:

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}[R \mid \mathbf{s}, \mathbf{a}, \pi]. \tag{4}$$

The best policy, given $Q^\pi(\mathbf{s}, \mathbf{a})$, can be found by choosing $\mathbf{a}$ greedily at every state: $\operatorname{argmax}_\mathbf{a} Q^\pi(\mathbf{s}, \mathbf{a})$. Under this policy, we can also define $V^\pi(\mathbf{s})$ by maximizing $Q^\pi(\mathbf{s}, \mathbf{a})$: $V^\pi(\mathbf{s}) = \max_\mathbf{a} Q^\pi(\mathbf{s}, \mathbf{a})$.

Dynamic programming

To actually learn Q^π, we exploit the Markov property and define the function as a Bellman equation [6], which has the following recursive form:

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\mathbf{s}_{t+1}}[r_{t+1} + \gamma Q^\pi(\mathbf{s}_{t+1}, \pi(\mathbf{s}_{t+1}))]. \tag{5}$$

This means that Q^π can be improved by bootstrapping, i.e., we can use the current values of our estimate of Q^π to improve our estimate. This is the foundation of Q-learning [94] and the state-action-reward-state-action (SARSA) algorithm [62]:

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q^\pi(\mathbf{s}_t, \mathbf{a}_t) + \alpha\delta, \tag{6}$$

where α is the learning rate and $\delta = Y - Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$ the temporal difference (TD) error; here, Y is a target as in a standard regression problem. SARSA, an on-policy learning algorithm, is used to improve the estimate of Q^π by using transitions generated by the behavioral policy (the policy derived from Q^π), which results in setting $Y = r_t + \gamma Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$. Q-learning

也可以考虑非情节性 MDP，其中 $T = \infty$。在这种情况下，$\gamma < 1$ 防止了奖励的无穷和被累积。此外，依赖于完整轨迹的方法不再适用，但使用有限转换集的方法仍然适用。

强化学习（RL）的一个关键概念是马尔可夫属性——只有当前状态会影响下一个状态，换句话说，在当前状态下，未来在某种程度上独立于过去。这意味着在任何状态下做出的决定 $\mathbf{s}_t$ 可以完全基于 $\mathbf{s}_{t+1}$，而不是 $\{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{t-1}\}$。尽管这个假设被大多数强化学习算法所接受，但它有些不现实，因为它要求状态是完全可观察的。MDP 的一种推广是部分可观察的 MDP（POMDP），其中代理接收到一个观察 $\mathbf{o}_t \in \Omega$，其中观察的分布 $p(\mathbf{o}_{t+1} \mid \mathbf{s}_{t+1}, \mathbf{a}_t)$ 取决于当前状态和先前的动作 [27]。在控制和信号处理上下文中，观察将由一个依赖于当前状态和先前应用的动作的状态空间模型中的测量 / 观察映射来描述。

POMDP 算法通常在给定先验信念状态、采取的动作和当前观测的情况下，维护对当前状态的信念。深度学习中更常见的方法是利用循环神经网络（RNNs）[20], [21], [48], [96], 与前馈神经网络不同，RNNs 是动态系统。

强化学习中的挑战

强调强化学习中面临的一些挑战是有益的：

- 最优策略必须通过与环境的试错交互来推断。智能体接收到的唯一学习信号是奖励。
- 智能体的观测依赖于其动作，并可能包含强的时间相关性。
- 智能体必须处理长程时间依赖性：通常，一个动作的后果只有在环境经过多次转换后才显现。这被称为（时间）信用分配问题 [78]。

我们将以室内机器人视觉导航任务为例来说明这些挑战：如果指定了目标位置，我们可能能够估计剩余距离（并将其用作奖励信号），但不太可能确切知道机器人需要采取哪些动作才能到达目标。随着机器人在建筑物中导航，它必须选择去哪里，其决策会影响它看到的房间，从而影响捕获的视觉序列的统计特性。最后，在导航了几个交叉口后，机器人可能会发现自己陷入死胡同。从学习动作的后果到平衡探索与利用，存在一系列问题，但最终这些都可以在强化学习框架内正式解决。

RL 算法

到目前为止，我们已经介绍了 RL 中使用的关键形式化方法，即 MDP，并简要提到了 RL 中的一些挑战。在接下来的内容中，我们将区分不同类别的 RL 算法。

解决 RL 问题的两种主要方法是：基于值函数的方法和基于策略搜索的方法。还有一种混合的 Actor-Critic 方法，它同时使用值函数和策略搜索。接下来，我们将解释这些方法以及解决 RL 问题时其他有用的概念。

Value functions

值函数方法基于估计处于给定状态的价值（预期回报）。状态值函数 $V^\pi(\mathbf{s})$ 是在状态 $\mathbf{s}$ 开始并随后遵循 π 时的预期回报：

$$V^\pi(\mathbf{s}) = \mathbb{E}[R \mid \mathbf{s}, \pi]. \tag{2}$$

最优策略，π^* 有对应的状态值函数 $V^*(\mathbf{s})$，反之亦然；最优状态值函数可以定义为

$$V^*(\mathbf{s}) = \max_\pi V^\pi(\mathbf{s}) \quad \forall \mathbf{s} \in \mathcal{S}. \tag{3}$$

如果我们有 $V^*(\mathbf{s})$ 可用，最优策略可以通过在 $\mathbf{s}_t$ 处所有可用的动作中选择，并选择最大化 $\mathbf{a}$ 的动作 $\mathbb{E}_{\mathbf{s}_{t+1} \sim \mathcal{T}(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a})}[V^*(\mathbf{s}_{t+1})]$。

在强化学习（RL）设置中，转移动态 T 不可用。因此，我们构建了另一个函数，即状态 - 动作值或质量函数 $Q^\pi(\mathbf{s}, \mathbf{a})$，它与 V^π 类似，但初始动作 $\mathbf{a}$ 已提供，而 π 仅从后续状态开始跟随：

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}[R \mid \mathbf{s}, \mathbf{a}, \pi]. \tag{4}$$

最佳策略，给定 $Q^\pi(\mathbf{s}, \mathbf{a})$，在状态 $\mathbf{s}$ 下可以通过贪婪地选择 $\mathbf{a}$ 在每个状态下：$\operatorname{argmax}_\mathbf{a} Q^\pi(\mathbf{s}, \mathbf{a})$。在此策略下，我们也可以通过最大化 $Q^\pi(\mathbf{s}, \mathbf{a})$ 来定义 $V^\pi(\mathbf{s}) = \max_\mathbf{a} Q^\pi(\mathbf{s}, \mathbf{a})$。

动态规划

要实际学习 Q^π，我们利用马尔可夫性质并将该函数定义为贝尔曼方程 [6]，其具有以下递归形式：

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\mathbf{s}_{t+1}}[r_{t+1} + \gamma Q^\pi(\mathbf{s}_{t+1}, \pi(\mathbf{s}_{t+1}))]. \tag{5}$$

这意味着 Q^π 可以通过自举来改进，即我们可以使用我们对 Q^π 的当前估计值来改进我们的估计。这是 Q- 学习 [94] 和状态 - 动作 - 奖励 - 状态 - 动作（SARSA）算法 [62] 的基础：

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q^\pi(\mathbf{s}_t, \mathbf{a}_t) + \alpha\delta, \tag{6}$$

其中 α 是学习率，$\delta = Y - Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$ 是时序差分 (TD) 误差；这里，Y 是一个标准回归问题中的目标。SARSA 是一种策略学习算法，用于通过行为策略（由导出的策略）生成的转换来改进 Q^π 的估计。这导致设置 $Y = r_t + \gamma Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$。$Q$- 学习

is off-policy, as Q^π is instead updated by transitions that were not necessarily generated by the derived policy. Instead, Q-learning uses $Y = r_t + \gamma \max_{\mathbf{a}} Q^\pi(\mathbf{s}_{t+1}, \mathbf{a})$, which directly approximates Q^*.

To find Q^* from an arbitrary Q^π, we use generalized policy iteration, where policy iteration consists of policy evaluation and policy improvement. Policy evaluation improves the estimate of the value function, which can be achieved by minimizing TD errors from trajectories experienced by following the policy. As the estimate improves, the policy can naturally be improved by choosing actions greedily based on the updated value function. Instead of performing these steps separately to convergence (as in policy iteration), generalized policy iteration allows for interleaving the steps, such that progress can be made more rapidly.

Sampling

Instead of bootstrapping value functions using dynamic programming methods, Monte Carlo methods estimate the expected return (2) from a state by averaging the return from multiple rollouts of a policy. Because of this, pure Monte Carlo methods can also be applied in non-Markovian environments. On the other hand, they can only be used in episodic MDPs, as a rollout has to terminate for the return to be calculated. It is possible to get the best of both methods by combining TD learning and Monte Carlo policy evaluation, as is done in the TD(λ) algorithm [78]. Similarly to the discount factor, the λ in TD(λ) is used to interpolate between Monte Carlo evaluation and bootstrapping. As demonstrated in Figure 3, this results in an entire spectrum of RL methods based around the amount of sampling utilized.

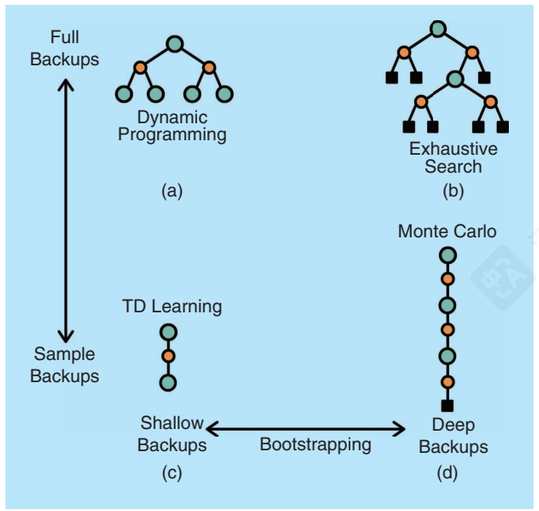


FIGURE 3. Two dimensions of RL algorithms based on the backups used to learn or construct a policy. At the extremes of these dimensions are (a) dynamic programming, (b) exhaustive search, (c) one-step TD learning, and (d) Monte Carlo approaches. Bootstrapping extends from (c) one-step TD learning to n-step TD learning methods [78], with (d) pure Monte Carlo approaches not relying on bootstrapping at all. Another possible dimension of variation is (c) and (d) choosing to sample actions versus (a) and (b) taking the expectation over all choices. (Figure recreated based on [78].)

Another major value-function-based method relies on learning the advantage function $A^\pi(\mathbf{s}, \mathbf{a})$ [3]. Unlike producing absolute state-action values, as with Q^π, A^π instead represents relative state-action values. Learning relative values is akin to removing a baseline or average level of a signal; more intuitively, it is easier to learn that one action has better consequences than another than it is to learn the actual return from taking the action. A^π represents a relative advantage of actions through the simple relationship $A^\pi = Q^\pi - V^\pi$ and is also closely related to the baseline method of variance reduction within gradient-based policy search methods [97]. The idea of advantage updates has been utilized in many recent DRL algorithms [19], [48], [71], [92].

Policy search

Policy search methods do not need to maintain a value function model but directly search for an optimal policy π^*. Typically, a parameterized policy π_θ is chosen, whose parameters are updated to maximize the expected return $\mathbb{E}[R|\theta]$ using either gradient-based or gradient-free optimization [12]. Neural networks that encode policies have been successfully trained using both gradient-free [17], [33] and gradient-based [22], [41], [44], [70], [71], [96], [97] methods. Gradient-free optimization can effectively cover low-dimensional parameter spaces, but, despite some successes in applying them to large networks [33], gradient-based training remains the method of choice for most DRL algorithms, being more sample efficient when policies possess a large number of parameters.

When constructing the policy directly, it is common to output parameters for a probability distribution; for continuous actions, this could be the mean and standard deviations of Gaussian distributions, while for discrete actions this could be the individual probabilities of a multinomial distribution. The result is a stochastic policy from which we can directly sample actions. With gradient-free methods, finding better policies requires a heuristic search across a predefined class of models. Methods such as evolution strategies essentially perform hill climbing in a subspace of policies [65], while more complex methods, such as compressed network search, impose additional inductive biases [33]. Perhaps the greatest advantage of gradient-free policy search is that it can also optimize nondifferentiable policies.

Policy gradients

Gradients can provide a strong learning signal as to how to improve a parameterized policy. However, to compute the expected return (1) we need to average over plausible trajectories induced by the current policy parameterization. This averaging requires either deterministic approximations (e.g., linearization) or stochastic approximations via sampling [12]. Deterministic approximations can be only applied in a model-based setting where a model of the underlying transition dynamics is available. In the more common model-free RL setting, a Monte Carlo estimate of the expected return is

是离策略的，因为 Q^π 是由那些并非由导出策略必然生成的状态转移来更新的。相反，Q- 学习使用 (,)max $Y r$ $Q\mathbf{s}\mathbf{a}_{r\ t\ \text{ka}\gamma} =\ +\ \ \ ^{\pi}\ _{+}$ 直接逼近 。$^{*}Q$

要找到 Q^* 从一个任意的 $,Q$ 我们使用广义策略迭代，其中策略迭代包括策略评估和策略改进。策略评估改进了值函数的估计，这可以通过最小化遵循策略所经历轨迹的 TD 误差来实现。随着估计的改进，策略可以通过根据更新的值函数贪婪地选择动作来自然地改进。而不是像策略迭代那样分别执行这些步骤直至收敛，广义策略迭代允许这些步骤交替进行，从而可以更快地取得进展。

Sampling

与其使用动态规划方法自举值函数，蒙特卡洛方法通过平均策略多次运行（ rollout ）的回报来估计状态（ 2 ）的预期回报。由于这个原因，纯蒙特卡洛方法也可以应用于非马尔可夫环境。另一方面，它们只能用于阶段式 MDP，因为运行（ rollout ）必须终止才能计算回报。通过结合 TD 学习和蒙特卡洛策略评估，可以得到两种方法的最佳效果，就像 TD([78]) 算法中做的那样。类似于折扣因子，TD({v6}) 中的 λ 用于在蒙特卡洛评估和自举之间进行插值。如图 3 所示，这导致

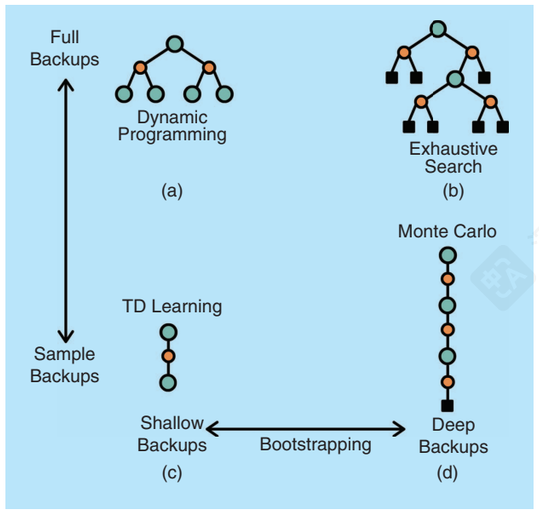


图 3。 基于用于学习或构建策略的回溯所使用的 RL 算法的两个维度。这些维度的两端分别是（ a ）动态规划，（ b ）穷举搜索，（ c ）单步 TD 学习和（ d ）蒙特卡洛方法。回溯从（ c ）单步 TD 学习扩展到 n- 步 TD 学习方法 [78], ，以及（ d ）完全不依赖回溯的纯蒙特卡洛方法。另一个可能的变体维度是（ c ）和（ d ）选择采样动作，而不是（ a ）和（ b ）对所有选择取期望。（图基于 [78] 重新创建。）

基于所利用的采样量的整个 RL 方法谱。

另一种基于值函数的主要方法依赖于学习优势函数 (,)A $\mathbf{sa}^{\pi}$ [3]。与生成绝对状态 - 动作值（如 $,QA$ $^{\pi}\ \ ^{\pi}$ ）不同，优势函数（ A ）表示相对状态 - 动作值。学习相对值类似于去除信号的一个基线或平均水平；更直观地说，学习一个动作比另一个动作有更好的结果比学习采取该动作的实际回报更容易。 A^π 通过简单的关系 A=Q - V^{π} $^{\pi}$ 表示动作的相对优势，并且也与基于梯度的策略搜索方法中的方差减少的基线方法密切相关 [97]。优势更新的思想已被许多最近的 DRL 算法利用 [19], [48], [71], [92]。

策略搜索

策略搜索方法不需要维护一个值函数模型，而是直接搜索最优策略。$^{*}\{\mathrm{r}\}_{\theta}$ 通常选择一个参数化的策略 $\{\mathrm{r}\}[\ \ |\ \ ]$ R{E}{i}，其参数通过基于梯度或无梯度的优化方法更新以最大化预期回报 [12]。编码策略的神经网络已成功使用无梯度 [17], [33] 和基于梯度 [22], [41], [44], [70], [71], [96], [97] 的方法进行训练。无梯度优化可以有效地覆盖低维参数空间，但在将它们应用于大型网络 [33], 方面取得了一些成功，基于梯度的训练仍然是大多数深度强化学习算法的首选方法，当策略具有大量参数时，它更有效率。

在直接构建策略时，通常输出概率分布的参数；对于连续动作，这可以是高斯分布的均值和标准差，而对于离散动作，这可以是多项式分布的各个概率。结果是我们可以从中直接采样动作的随机策略。使用无梯度方法，找到更好的策略需要在预定义的模型类中进行启发式搜索。进化策略等方法本质上在策略的子空间中执行爬山 [65], ，而更复杂的方法，如压缩网络搜索，则施加了额外的归纳偏差 [33]。无梯度策略搜索的最大优势在于它还可以优化不可微分的策略。

Policy gradients

梯度可以为如何改进参数化策略提供强有力的学习信号。然而，为了计算预期回报（ 1 ），我们需要对当前策略参数化所诱导的合理轨迹进行平均。这种平均需要要么使用确定性近似（例如，线性化），要么通过采样 [12] 进行随机近似。确定性近似只能应用于基于模型的环境，其中可获取底层转换动态的模型。在更常见的无模型 RL 设置中，预期回报的蒙特卡洛估计是

determined. For gradient-based learning, this Monte Carlo approximation poses a challenge since gradients cannot pass through these samples of a stochastic function. Therefore, we turn to an estimator of the gradient, known in RL as the REINFORCE rule [97]. Intuitively, gradient ascent using the estimator increases the log probability of the sampled action, weighted by the return. More formally, the REINFORCE rule can be used to compute the gradient of an expectation over a function f of a random variable X with respect to parameters θ:

$$\nabla_\theta \mathbb{E}_X[f(X;\theta)] = \mathbb{E}_X[f(X;\theta)\nabla_\theta \log p(X)]. \quad (7)$$

As this computation relies on the empirical return of a trajectory, the resulting gradients possess a high variance. By introducing unbiased estimates that are less noisy, it is possible to reduce the variance. The general methodology for performing this is to subtract a baseline, which means weighting updates by an advantage rather than the pure return. The simplest baseline is the average return taken over several episodes [97], but there are many more options available [71].

Actor-critic methods

It is possible to combine value functions with an explicit representation of the policy, resulting in actor-critic methods, as shown in Figure 4. The "actor" (policy) learns by using feedback from the "critic" (value function). In doing so, these methods tradeoff variance reduction of policy gradients with bias introduction from value function methods [32], [71].

Actor-critic methods use the value function as a baseline for policy gradients, such that the only fundamental difference between actor-critic methods and other baseline methods is that actor-critic methods utilize a learned value function. For this reason, we will later discuss actor-critic methods as a subset of policy gradient methods.

Planning and learning

Given a model of the environment, it is possible to use dynamic programming over all possible actions [Figure 3(a)], sample trajectories for heuristic search (as was done by AlphaGo [73]), or even perform an exhaustive search [Figure 3(b)]. Sutton and Barto [78] define *planning* as any method that utilizes a model to produce or improve a policy. This includes distribution models, which include $\mathcal{T}$ and $\mathcal{R}$, and sample models, from which only samples of transitions can be drawn.

In RL, we focus on learning without access to the underlying model of the environment. However, interactions with the environment could be used to learn value functions, policies, and also a model. Model-free RL methods learn directly from interactions with the environment, but model-based RL methods can simulate transitions using the learned model, resulting in increased sample efficiency. This is particularly important in domains where each interaction with the environment is expensive. However, learning a model introduces extra complexities, and there is always the danger of suffering from model errors, which in turn affects the learned policy. Although deep neural networks can potentially produce very complex and rich models [14], [55], [75], sometimes simpler, more data-efficient methods are preferable [19]. These considerations also play a role in actor-critic methods with learned value functions [32], [71].

Searching directly for a policy represented by a neural network with very many parameters can be difficult and can suffer from severe local minima.

The rise of DRL

Many of the successes in DRL have been based on scaling up prior work in RL to high-dimensional problems. This is due to the learning of low-dimensional feature representations and the powerful function approximation properties of neural networks. By means of representation learning, DRL can deal efficiently with the curse of dimensionality, unlike tabular and traditional nonparametric methods [7]. For instance, convolutional neural networks (CNNs) can be used as components of RL agents, allowing them to learn directly from raw, high-dimensional visual inputs. In general, DRL is based on training deep neural networks to approximate the optimal policy π^* and/or the optimal value functions V^*, Q^*, and A^*.

Value functions

The well-known function approximation properties of neural networks led naturally to the use of deep learning to regress functions for use in RL agents. Indeed, one of the earliest success stories in RL is TD-Gammon, a neural network that reached expert-level performance in backgammon in the early 1990s [81]. Using TD methods, the network took in the state of the board to predict the probability of black or white winning. Although this simple idea has been echoed in later work [73], progress in RL research has favored the explicit use of value functions, which can capture the

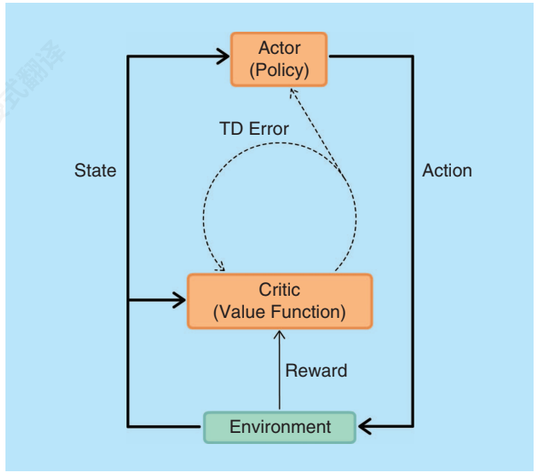


FIGURE 4. The actor-critic setup. The actor (policy) receives a state from the environment and chooses an action to perform. At the same time, the critic (value function) receives the state and reward resulting from the previous interaction. The critic uses the TD error calculated from this information to update itself and the actor. (Figure recreated based on [78].)

确定的。对于基于梯度的学习，这种蒙特卡洛近似提出了挑战，因为梯度无法通过随机函数的这些样本传递。因此，我们转向梯度估计器，在强化学习中称为 REINFORCE 规则 [97]。直观地讲，使用估计器进行梯度上升会增加采样动作的对数概率，并由回报加权。更多形式上，REINFORCE 规则可用于计算关于随机变量 X 的函数 f 的期望的梯度，其中 X 的参数为 i。

$$\nabla_\theta \mathbb{E}_X[f(X;\theta)] = \mathbb{E}_X[f(X;\theta)\nabla_\theta \log p(X)]. \quad (7)$$

由于此计算依赖于轨迹的经验回报，因此产生的梯度具有很高的方差。通过引入更少噪声的无偏估计，可以减少方差。执行此操作的通用方法是从中减去一个基线，这意味着通过优势而不是纯回报来加权更新。最简单的基线是跨多个回合 [97], 取的平均回报，但还有许多其他选项可用 [71]。

Actor-critic 方法

可以将值函数与策略的显式表示相结合，从而得到 Actor-critic 方法，如图 4 所示。“Actor”（策略）通过 “Critic”（值函数）的反馈进行学习。在此过程中，这些方法在策略梯度的方差减少和值函数方法引入的偏差之间进行了权衡 [32], [71]。

Actor-critic 方法使用值函数作为策略梯度的基线，因此 Actor-critic 方法与其他基线方法之间唯一的根本区别在于 Actor-critic 方法利用了学习到的值函数。因此，我们将在后面将 Actor-critic 方法讨论为策略梯度方法的一个子集。

规划与学习

给定环境的模型，可以使用动态规划对所有可能的动作 [图3(a)], 样本轨迹进行启发式搜索（如 AlphaGo[73] 所做的那样），甚至可以进行穷举搜索 [图 3(b)]。Sutton 和 Barto[78] 将规划定义为任何利用模型来生成或改进策略的方法。这包括分布模型，包括 T 和 R，以及样本模型，从中只能抽取转换的样本。

In RL, we focus on learning without access to the underlying model of the environment. However, interactions with the environment could be used to learn value functions, policies, and also a model. Model-free RL methods learn directly from interactions with the environment, but model-based RL methods can simulate transitions using the learned model, resulting in increased sample efficiency. This is particularly important in domains where each interaction with the environment is expensive. However, learning a model introduces extra complexities, and there is always the

直接搜索由参数量非常大的神经网络表示的策略可能很困难，并且可能遭受严重的局部最小值问题。

模型错误的风险，进而影响所学习的策略。尽管深度神经网络有可能产生非常复杂和丰富的模型 [14], [55], [75],，但有时更简单、更数据高效的方法更可取 [19]。这些考虑因素也适用于具有学习价值函数的 Actor-Critic 方法 [32], [71]。

The rise of DRL

许多 DRL 的成功都是基于将先前在 RL 中的工作扩展到高维问题。这是由于低维特征表示的学习和神经网络强大的函数逼近特性。通过表示学习，DRL 可以有效地处理维数灾难，这与表格和传统的非参数方法不同。例如，卷积神经网络（ CNN ）可以用作 RL 代理的组件，允许它们直接从原始的高维视觉输入中学习。一般来说，DRL 基于训练深度神经网络来逼近最优策略 r 和 / 或最优值函数 V, Q。

价值函数

神经网络众所周知的函数逼近特性自然而然地引出了使用深度学习来回归用于强化学习智能体的函数。事实上，强化学习中最早的成功故事之一是 TD-Gammon，一个在 20 世纪 90 年代初达到围棋专家级水平的神经网络 [81]。使用 TD 方法，网络接收棋盘状态来预测黑白双方获胜的概率。尽管这个简单的想法后来在研究中得到了呼应 [73],，但强化学习研究的进展更倾向于显式使用价值函数，这些函数可以捕捉

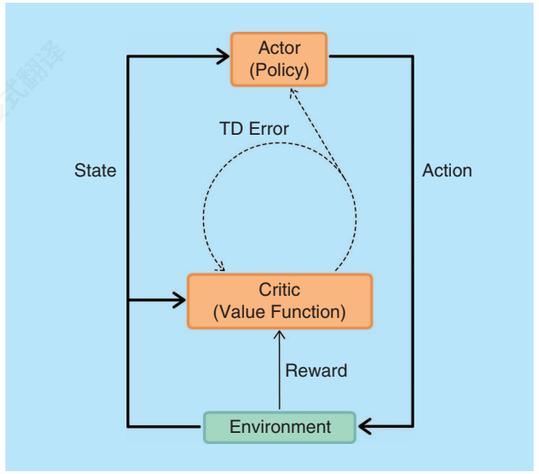


图 4。 Actor-Critic 设置。Actor （策略）从环境中接收一个状态并选择一个要执行的动作。同时，Critic （价值函数）接收先前交互产生的状态和奖励。Critic 使用从这些信息中计算出的 TD 误差来更新自己和 Actor。（图基于 [78] 重新创建。）

structure underlying the environment. From early value function methods in DRL, which took simple states as input [61], current methods are now able to tackle visually and conceptually complex environments [47], [48], [70], [100].

Function approximation and the DQN

We begin our survey of value-function-based DRL algorithms with the DQN [47], illustrated in Figure 5, which achieved scores across a wide range of classic Atari 2600 video games [5] that were comparable to that of a professional video games tester. The inputs to the DQN are four gray-scale frames of the game, concatenated over time, which are initially processed by several convolutional layers to extract spatiotemporal features, such as the movement of the ball in *Pong* or *Breakout*. The final feature map from the convolutional layers is processed by several fully connected layers, which more implicitly encode the effects of actions. This contrasts with more traditional controllers that use fixed preprocessing steps, which are therefore unable to adapt their processing of the state in response to the learning signal.

A forerunner of the DQN—neural-fitted Q (NFQ) iteration—involved training a neural network to return the Q-value given a state-action pair [61]. NFQ was later extended to train a network to drive a slot car using raw visual inputs from a camera over the race track, by combining a deep autoencoder to reduce the dimensionality of the inputs with a separate branch to predict Q-values [38]. Although the previous network could have been trained for both reconstruction and RL tasks simultaneously, it was both more reliable and computationally efficient to train the two parts of the network sequentially.

The DQN [47] is closely related to the model proposed by Lange et al. [38] but was the first RL algorithm that was demonstrated to work directly from raw visual inputs and on a wide variety of environments. It was designed such that the final fully connected layer outputs $Q^{\pi}(\mathbf{s}, \cdot)$ for all action values in a discrete set of actions—in this case, the various directions of the joystick and the fire button. This not only enables the best action, $\text{argmax}_{\mathbf{a}} Q^{\pi}(\mathbf{s}, \mathbf{a})$, to be chosen after a single forward pass of the network, but also allows the network to more easily encode action-independent knowledge in the lower, convolutional layers. With merely the goal of maximizing its score on a video game, the DQN learns to extract salient visual features, jointly encoding objects, their movements, and, most importantly, their interactions. Using techniques originally developed for explaining the behavior of CNNs in object recognition tasks, we can also inspect what parts of its view the agent considers important (see Figure 6).

The true underlying state of the game is contained within 128 bytes of Atari 2600 random-access memory. However, the DQN was designed to directly learn from visual inputs (210×160 pixel 8-bit RGB images), which it takes as the state $\mathbf{s}$. It is impractical to represent $Q^{\pi}(\mathbf{s}, \mathbf{a})$ exactly as a lookup table: when combined with 18 possible actions, we obtain a Q-table of size $|\mathcal{S}| \times |\mathcal{A}| = 18 \times 256^{3 \times 210 \times 160}$. Even if it were feasible to create such a table, it would be sparsely populated, and information gained from one state-action pair cannot be propagated to other state-action pairs. The strength of the DQN lies in its ability to compactly represent both high-dimensional observations and the Q-function using deep neural networks. Without this ability, tackling the discrete Atari domain from raw visual inputs would be impractical.

The DQN addressed the fundamental instability problem of using function approximation in RL [83] by the use of two techniques: experience replay [45] and target networks. Experience replay memory stores transitions of the form $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_{t+1})$ in a cyclic buffer, enabling the RL agent to sample from and train on previously observed data offline. Not only does this massively reduce the number of interactions needed with the environment, but batches of experience can be sampled, reducing the variance of learning updates. Furthermore, by sampling uniformly from a large memory, the temporal correlations that can adversely affect RL algorithms are broken. Finally, from a practical perspective, batches of data can be efficiently processed in parallel by modern hardware, increasing throughput. While the original DQN algorithm used uniform sampling [47], later work showed that prioritizing samples based on TD errors is more effective for learning [67].

The second stabilizing method, introduced by Mnih et al. [47], is the use of a target network that initially contains the weights of the network enacting the policy but is kept frozen for a large period of time. Rather than having to calculate the

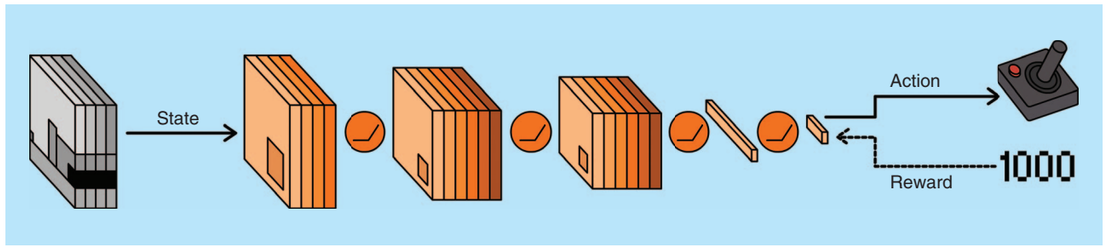


FIGURE 5. The DQN [47]. The network takes the state—a stack of gray-scale frames from the video game—and processes it with convolutional and fully connected layers, with ReLU nonlinearities in between each layer. At the final layer, the network outputs a discrete action, which corresponds to one of the possible control inputs for the game. Given the current state and chosen action, the game returns a new score. The DQN uses the reward—the difference between the new score and the previous one—to learn from its decision. More precisely, the reward is used to update its estimate of Q, and the error between its previous estimate and its new estimate is backpropagated through the network.

环境底层结构。从深度强化学习中的早期值函数方法，这些方法将简单状态作为输入 [61]，当前方法现在能够处理视觉和概念上复杂的环境 [47], [48], [70], [100]。

函数逼近和 *DQN*

我们从基于值函数的深度强化学习算法开始调查，以图 5 中所示的 DQN [47]，在一系列经典 Atari 2600 视频游戏中取得了与专业视频游戏测试员相当的成绩 [5]。DQN 的输入是四个灰度游戏帧，随时间连接，最初由几个卷积层处理以提取时空特征，例如球在 *Pong* 或 *Breakout* 中的运动。卷积层的最终特征图由几个全连接层处理，这些层更隐式地编码了动作的效果。这与使用固定预处理步骤的更传统控制器形成对比，因此它们无法根据学习信号调整其对状态的处理。

DQN 的先驱 — 神经拟合 Q (NFQ) 迭代 — 涉及训练一个神经网络来返回给定状态 - 动作对的 Q- 值 [61]。NFQ 后来被扩展到训练一个网络，使用来自赛道摄像头的原始视觉输入来驱动赛车，通过结合深度自动编码器来降低输入的维度，并使用一个单独的分支来预测 Q- 值 [38]。尽管之前的网络可以同时用于重建和强化学习任务，但按顺序训练网络的两个部分既更可靠又更计算高效。

DQN [47] 与 Lange 等人提出的模型密切相关 [38]，但它是第一个被证明可以直接从原始视觉输入并在各种环境中工作的强化学习算法。它的设计方式是，最终的全连接层输出 $(,)Q \mathbf{s}^{\pi}$ 对于离散动作集中的所有动作值 —— 在这种情况下，操纵杆的各种方向和发射按钮。这不仅使得在网络的单次前向传递后可以选择最佳动作，$\text{argmax}\ (\ ,\ )Q\ \mathbf{s}\ \mathbf{a}_{\mathbf{a}}^{\ \pi}$ ，还允许网络更容易地在较低、卷积层中编码与动作无关的知识。仅仅为了在最大化其在视频游戏上的得分时，DQN 学习提取显著的视觉特征，联合编码对象、它们的运动，以及最重要的是它们的交互。使用最初为解释对象识别任务中 CNN 行为而开发的技术，我们也可以检查智能体认为其视野中哪些部分是重要的（见图 6）。

游戏的真正底层状态包含在 Atari 2600 随机存取内存的 128 字节中。然而，DQN 被设计为直接从视觉输入中学习 (210160× 像素 8 位 RGB 图像)，它将其视为状态。$\mathbf{s}$ 表示 $(\ ,\ )Q\ \mathbf{s}\mathbf{a}^{\pi}$ 精确地作为一个查找表：当与 18 种可能的动作结合时，我们得到一个大小为 Q- 表。$18 \mid 256\mathcal{A}\mathcal{S}^{\ 3\ \ 210\ \ 160} \times \times = \quad \times \times$ 即使创建这样的表是可行的，它也会是稀疏的，并且从一对状态 - 动作中获得的信息不能传播到其他状态 - 动作对。DQN 的力量在于它能够使用深度神经网络紧凑地表示高维观察结果和 Q- 函数。没有这种能力，从原始视觉输入中处理离散的 Atari 域将是不切实际的。

DQN 通过使用函数逼近解决了 RL 中应用函数逼近的基本不稳定性问题 [83]，采用了两种技术：经验回放 [45] 和目标网络。经验回放存储形式为 (s, a, r, s)r **的转换** $_{t}\ _{t}\ _{t+1}\ _{t+1}$，以循环缓冲区的方式存储，使 RL 代理能够离线地从先前观察到的数据中采样并训练。这不仅极大地减少了与环境的交互次数，而且可以采样经验批次，减少学习更新的方差。此外，通过从大内存中均匀采样，可以打破可能对 RL 算法产生不利影响的时序相关性。最后，从实际角度来看，数据批次可以由现代硬件高效并行处理，从而提高吞吐量。虽然原始 DQN 算法使用均匀采样 [47],，但后续研究表明，根据 TD 误差对样本进行优先级排序更有利于学习 [67]。

第二种稳定方法由 Mnih 等人 .[47]，引入，是使用一个目标网络，该网络最初包含执行策略的网络权重，但在很长一段时间内保持冻结。而不是必须计算

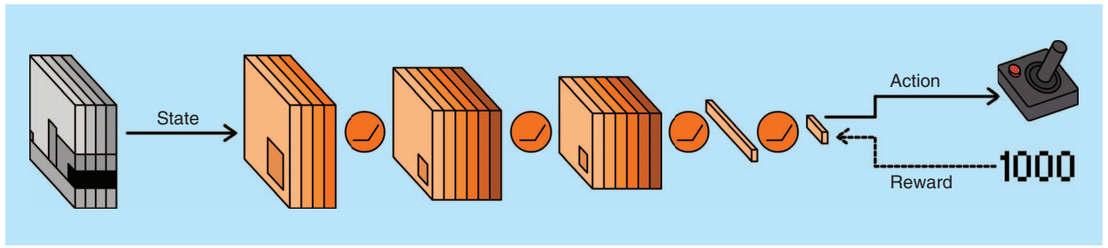


图 5. 深度 Q 网络 [47]。该网络接收状态 —— 视频游戏的一堆灰度帧 —— 并通过卷积层和全连接层进行处理，每层之间使用 ReLU 非线性。在最后一层，网络输出一个离散动作，该动作对应于游戏的一种可能控制输入。给定当前状态和选择的动作，游戏返回一个新的分数。深度 Q 网络使用奖励 —— 新分数与旧分数之间的差异 —— 来学习其决策。更精确地说，奖励用于更新其对 $,Q$ 的估计，并且其先前估计与其新估计之间的误差通过网络反向传播。

TD error based on its own rapidly fluctuating estimates of the Q-values, the policy network uses the fixed target network. During training, the weights of the target network are updated to match the policy network after a fixed number of steps. Both experience replay and target networks have gone on to be used in subsequent DRL works [19], [44], [50], [93].

Q-function modifications

Considering that one of the key components of the DQN is a function approximator for the Q-function, it can benefit from fundamental advances in RL. In [86], van Hasselt showed that the single estimator used in the Q-learning update rule overestimates the expected return due to the use of the maximum action value as an approximation of the maximum expected action value. Double-Q learning provides a better estimate through the use of a double estimator [86]. While double-Q learning requires an additional function to be learned, later work proposed using the already available target network from the DQN algorithm, resulting in significantly better results with only a small change in the update step [87].

Yet another way to adjust the DQN architecture is to decompose the Q-function into meaningful functions, such as constructing Q^π by adding together separate layers that compute the state-value function V^π and advantage function A^π [92]. Rather than having to come up with accurate Q-values for all actions, the duelling DQN [92] benefits from a single baseline for the state in the form of V^π and easier-to-learn relative values in the form of A^π. The combination of the duelling DQN with prioritized experience replay [67] is one of the state-of-the-art techniques in discrete action settings. Further insight into the properties of A^π by Gu et al. [19] led them to modify the DQN with a convex advantage layer that extended the algorithm to work over sets of continuous actions, creating the normalized advantage function (NAF) algorithm. Benefiting from experience replay, target networks, and advantage updates, NAF is one of several state-of-the-art techniques in continuous control problems [19].

Policy search

Policy search methods aim to directly find policies by means of gradient-free or gradient-based methods. Prior to the current surge of interest in DRL, several successful methods in DRL eschewed the commonly used backpropagation algorithm in favor of evolutionary algorithms [17], [33], which are gradient-free policy search algorithms. Evolutionary methods rely on evaluating the performance of a population of agents. Hence, they are expensive for large populations or agents with many parameters. However, as black-box optimization methods, they can be used to optimize arbitrary, nondifferentiable models and naturally allow for more exploration in the parameter space. In combination with a compressed representation of neural network weights, evolutionary algorithms can even be used to train large networks; such a technique resulted in the first deep neural network to learn an RL task, straight from high-dimensional visual inputs [33]. Recent work has reignited interest in evolutionary methods for RL as they can

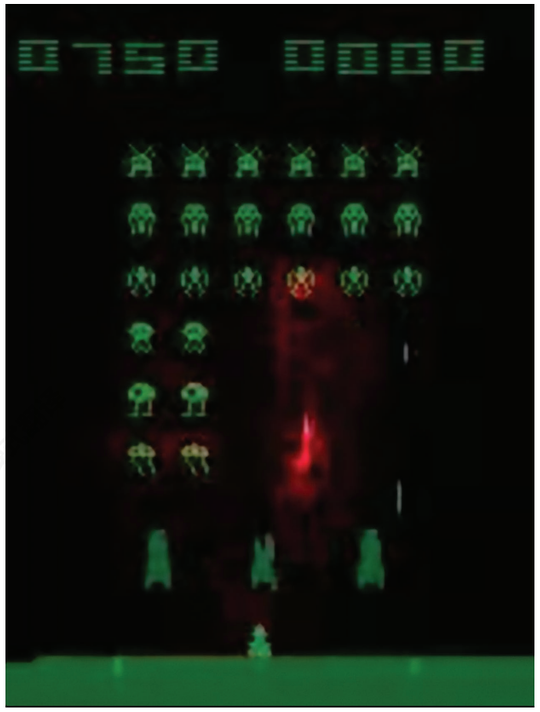


FIGURE 6. A saliency map of a trained DQN [47] playing *Space Invaders* [5]. By backpropagating the training signal to the image space, it is possible to see what a neural-network-based agent is attending to. In this frame, the most salient points—shown with the red overlay—are the laser that the agent recently fired and also the enemy that it anticipates hitting in a few time steps.

potentially be distributed at larger scales than techniques that rely on gradients [65].

Backpropagation through stochastic functions

The workhorse of DRL, however, remains backpropagation. The previously discussed REINFORCE rule [97] allows neural networks to learn stochastic policies in a task-dependent manner, such as deciding where to look in an image to track [69] or caption [99] objects. In these cases, the stochastic variable would determine the coordinates of a small crop of the image and hence reduce the amount of computation needed. This usage of RL to make discrete, stochastic decisions over inputs is known in the deep-learning literature as *hard attention* and is one of the more compelling uses of basic policysearch methods in recent years, having many applications outside of traditional RL domains.

Compounding errors

Searching directly for a policy represented by a neural network with very many parameters can be difficult and suffer from severe local minima. One way around this is to use guided policy search (GPS), which takes a few sequences of actions from another controller (which could be constructed using a separate method, such

TD 误差基于其自身快速波动的估计值，策略网络使用固定目标网络。在训练过程中，目标网络的权重在固定步数后更新以匹配策略网络。经验回放和目标网络都已经在后续的深度强化学习工作中得到应用 Q-值， 策略网络使用固定目标网络。在训练过程中，目标网络的权重在固定步数后更新以匹配策略网络。经验回放和目标网络都已经在后续的深度强化学习工作中得到应用 [19], [44], [50], [93]。

Q 函数修改

考虑到 DQN 的一个关键组件是 Q 函数的函数逼近器，它可以受益于强化学习的基本进展。在 Q- 函数，它可以受益于强化学习的基本进展。在 [86], van Hasselt 表明，在 Q- 学习更新规则中使用的单个估计器由于使用最大动作值作为最大预期动作值的近似值而高估了预期回报。双 Q 学习通过使用双估计器提供了更好的估计 [86]。虽然双 Q 学习需要一个额外的函数来学习，但后续工作提出了使用 DQN 算法中已经可用的目标网络，仅在更新步骤中做了微小改变，但结果显著提高 [87]。

另一种调整 DQN 架构的方法是将 Q- 函数分解为有意义的函数，例如通过将计算状态值函数 Q^π 和优势函数 V^π A$^\pi$[92] 的各个层相加来构建 Q- 值，而不是为所有动作都提出准确的 Q- 值，*duelling DQN* [92] 受益于以 V$^\pi$ 和 <style id='39'>A$^\pi$ 形式存在的状态的单个基线，以及更容易学习的以 形式存在的相对值 A^π 。duelling DQN 与优先经验回放的结合是离散动作设置中的最先进技术之一。Gu 等人对 A[19] 的属性进行的进一步研究，促使他们通过添加一个凸优势层来修改 DQN，将算法扩展到连续动作集上，创建了归一化优势函数（ NAF ）算法。受益于经验回放、目标网络和优势更新，NAF 是连续控制问题中的几种最先进技术之一

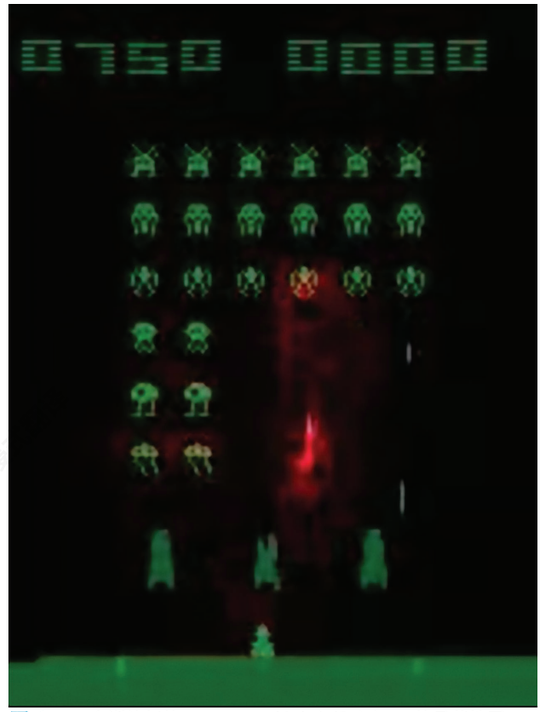


图 6。 一个训练好的 DQN [47] 玩 太空侵略者 [5] 的显著性图。通过将训练信号反向传播到图像空间，可以看到基于神经网络的智能体在关注什么。在这个帧中，最显著的点 —— 用红色叠加显示的 —— 是智能体最近发射的激光，以及它预期在几个时间步内击中的敌人。

在比依赖梯度的技术 [65] 更大的规模上潜在地分布。

Backpropagation through stochastic functions

然而，深度强化学习的核心仍然是反向传播。之前讨论的 REINFORCE 规则 [97] 允许神经网络以任务相关的方式学习随机策略，例如决定在图像中注视何处以跟踪 [69] 或描述 [99] 对象。在这些情况下，随机变量将决定图像小裁剪区域的坐标，从而减少所需的计算量。这种使用强化学习对输入进行离散、随机决策的方式在深度学习文献中被称为硬注意力，并且是近年来基本策略搜索方法更具吸引力的应用之一，在传统的强化学习领域之外也有许多应用。

Compounding errors

直接搜索由具有非常多参数的神经网络表示的策略可能很困难，并且会遭受严重的局部最小值。解决这个问题的一种方法是用引导策略搜索（ GPS ），它从另一个控制器（该控制器可以使用单独的方法构建，例如

策略搜索

策略搜索方法旨在通过无梯度或基于梯度的方法直接找到策略。在深度强化学习（ DRL ）当前兴趣激增之前，DRL 中有几种成功的方法放弃了常用的反向传播算法，而选择了进化算法 [17], [33], ，这些是无梯度的策略搜索算法。进化方法依赖于评估一组智能体的性能。因此，对于大型种群或具有许多参数的智能体来说，它们是昂贵的。然而，作为黑盒优化方法，它们可以用于优化任意、不可微分的模型，并自然地允许在参数空间中进行更多探索。结合神经网络权重的压缩表示，进化算法甚至可以用于训练大型网络；这种技术导致了第一个深度神经网络直接从高维视觉输入中学习强化学习任务 [33]。最近的研究重新点燃了对进化方法在强化学习中的兴趣，因为它们可以

as optimal control). GPS learns from them by using supervised learning in combination with importance sampling, which corrects for off-policy samples [40]. This approach effectively biases the search toward a good (local) optimum. GPS works in a loop, by optimizing policies to match sampled trajectories and optimizing trajectory distributions to match the policy and minimize costs. Levine et al. [41] showed that it was possible to train visuomotor policies for a robot "end to end," straight from the RGB pixels of the camera to motor torques, and, hence, provide one of the seminal works in DRL.

A more commonly used method is to use a trust region, in which optimization steps are restricted to lie within a region where the approximation of the true cost function still holds. By preventing updated policies from deviating too wildly from previous policies, the chance of a catastrophically bad update is lessened, and many algorithms that use trust regions guarantee or practically result in monotonic improvement in policy performance. The idea of constraining each policy gradient update, as measured by the Kullback–Leibler (KL) divergence between the current and proposed policy, has a long history in RL [28]. One of the newer algorithms in this line of work, TRPO, has been shown to be relatively robust and applicable to domains with high-dimensional inputs [70]. To achieve this, TRPO optimizes a surrogate objective function—specifically, it optimizes an (importance sampled) advantage estimate, constrained using a quadratic approximation of the KL divergence. While TRPO can be used as a pure policy gradient method with a simple baseline, later work by Schulman et al. [71] introduced generalized advantage estimation (GAE), which proposed several, more advanced variance reduction baselines. The combination of TRPO and GAE remains one of the state-of-the-art RL techniques in continuous control.

Actor-critic methods

Actor-critic approaches have grown in popularity as an effective means of combining the benefits of policy search methods with learned value functions, which are able to learn from full returns and/or TD errors. They can benefit from improvements in both policy gradient methods, such as GAE [71], and value function methods, such as target networks [47]. In the last few years, DRL actor-critic methods have been scaled up from learning simulated physics tasks [22], [44] to real robotic visual navigation tasks [100], directly from image pixels.

One recent development in the context of actor-critic algorithms is deterministic policy gradients (DPGs) [72], which extend the standard policy gradient theorems for stochastic policies [97] to deterministic policies. One of the major advantages of DPGs is that, while stochastic policy gradients integrate over both state and action spaces, DPGs only integrate over the state space, requiring fewer samples in problems with large action spaces. In the initial work on DPGs, Silver et al. [72] introduced and demonstrated an off-policy actor-critic algorithm that vastly improved upon a stochastic policy gradient equivalent in high-dimensional continuous control problems. Later work introduced deep DPG, which utilized neural networks to operate on high-dimensional, visual state spaces [44]. In the same vein as DPGs, Heess et al. [22] devised a method for calculating gradients to optimize stochastic policies by "reparameterizing" [30], [60] the stochasticity away from the network, thereby allowing standard gradients to be used (instead of the high-variance REINFORCE estimator [97]). The resulting stochastic value gradient (SVG) methods are flexible and can be used both with (SVG(0) and SVG(1)) and without (SVG(∞)) value function critics, and with (SVG(∞) and SVG(1)) and without (SVG(0)) models. Later work proceeded to integrate DPGs and SVGs with RNNs, allowing them to solve continuous control problems in POMDPs, learning directly from pixels [21]. Together, DPGs and SVGs can be considered algorithmic approaches for improving learning efficiency in DRL.

An orthogonal approach to speeding up learning is to exploit parallel computation. By keeping a canonical set of parameters that are read by and updated in an asynchronous fashion by multiple copies of a single network, computation can be efficiently distributed over both processing cores in a single central processing unit (CPU), and across CPUs in a cluster of machines. Using a distributed system, Nair et al. [51] developed a framework for training multiple DQNs in parallel, achieving both better performance and a reduction in training time. However, the simpler asynchronous advantage actor-critic (A3C) algorithm [48], developed for both single and distributed machine settings, has become one of the most popular DRL techniques in recent times. A3C combines advantage updates with the actor-critic formulation and relies on asynchronously updated policy and value function networks trained in parallel over several processing threads. The use of multiple agents, situated in their own, independent environments, not only stabilizes improvements in the parameters, but conveys an additional benefit in allowing for more exploration to occur. A3C has been used as a standard starting point in many subsequent works, including the work of Zhu et al. [100], who applied it to robotic navigation in the real world through visual inputs.

There have been several major advancements on the original A3C algorithm that reflect various motivations in the field of DRL. The first is actor-critic with experience replay [93], which adds off-policy bias correction to A3C, allowing it to use experience replay to improve sample complexity. Others have attempted to bridge the gap between value and policy-based RL, utilizing theoretical advancements to improve upon the original A3C [50], [54]. Finally, there is a growing trend toward exploiting auxiliary tasks to improve the representations learned by DRL agents and, hence, improve both the learning speed and final performance of these agents [26], [46].

Current research and challenges

To conclude, we will highlight some current areas of research in DRL and the challenges that still remain. Previously, we have focused mainly on model-free methods, but we will now examine a few model-based DRL algorithms in more detail. Model-based RL algorithms play an important role in making RL data efficient and in trading off exploration and exploitation. After tackling exploration strategies, we shall then address hierarchical

作为最优控制）。GPS 通过结合监督学习和重要性采样来学习它们，这可以纠正离线策略样本 [40]。这种方法有效地使搜索偏向于一个良好的（局部）最优解。GPS 在一个循环中工作，通过优化策略以匹配采样轨迹，并优化轨迹分布以匹配策略并最小化成本。Levine 等人 [41] 证明了可以为机器人 " 端到端 " 地训练视觉运动策略，直接从相机的 RGB 像素到电机扭矩，从而为深度强化学习（ DRL ）提供了开创性的工作。

一种更常用的方法是使用信任域，其中优化步骤被限制在一个区域内，该区域内的真实成本函数近似仍然成立。通过防止更新策略与先前策略偏差过大，灾难性坏更新的可能性就会降低，并且许多使用信任域的算法保证或实际上导致策略性能单调改进。在强化学习（ RL ）中，约束每个策略梯度更新，通过当前策略和提议策略之间的 Kullback-Leibler (KL) 散度来衡量，这一想法有着悠久的历史 [28]。在这项工作中较新的算法 TRPO 已被证明相对鲁棒，并适用于具有高维输入的领域 [70]。为此，TRPO 优化了一个替代目标函数 —— 具体来说，它优化了一个（重要性采样）优势估计，并使用 KL 散度的二次近似进行约束。虽然 TRPO 可以作为具有简单基线的纯策略梯度方法使用，但 Schulman 等人 [71] 的后续工作引入了广义优势估计（ GAE ），该估计提出了几个更先进的方差减少基线。TRPO 和 GAE 的结合仍然是连续控制领域中的一种最先进的强化学习技术。

Actor-critic 方法

Actor-critic 方法作为一种有效结合策略搜索方法与学习价值函数的手段而日益流行，这些方法能够从完整回报和 / 或 TD 误差中学习。它们可以从策略梯度方法（如 GAE [71], ）和价值函数方法（如目标网络 [47]）的改进中受益。在过去的几年里，深度强化学习（ DRL ）的 Actor-critic 方法已经从学习模拟物理任务 [22], [44] 扩展到直接从图像像素中学习真实机器人视觉导航任务 [100], 。

Actor-critic 算法的一个最新发展是确定性策略梯度（ DPGs ） [72], ，它将标准的策略梯度定理从随机策略 [97] 扩展到确定性策略。DPGs 的主要优势在于，虽然随机策略梯度在状态空间和动作空间上积分，但 DPGs 只在状态空间上积分，在具有大动作空间的问题中需要更少的样本。在 DPGs 的初始工作中，Silver 等人 [72] 引入并展示了一种离线策略 Actor-critic 算法，该算法在高维连续控制问题中大大优于随机策略梯度等效算法。后来的工作引入了深度 DPG，它利用神经网络来处理高维数据，视觉状态空间 [44]。与 DPG 类似，Heess 等人 [22] 设计了一种通过 " 重新参数化 " [30], [60] 将随机性从网络中分离出来以计算梯度来优化随机策略的方法，从而可以使用标准梯度（而不是高方差的 REINFORCE 估计器 [97] ）。由此产生的随机值梯度 (SVG) 方法是灵活的，可以与（ SVG(0) 和 SVG(1) ）以及不与（ SVG(∞)) 值函数评论家一起使用，以及与（ SVG (∞) 和 SVG(1) ）以及不与（ SVG(0) ）模型一起使用。后来的工作继续将 DPG 和 SVG 与 RNN 集成，使它们能够解决 POMDP 中的连续控制问题，直接从像素 [21] 学习。总而言之，DPG 和 SVG 可以被认为是用于提高深度强化学习 (DRL) 中学习效率的算法方法。

一种加快学习速度的正交方法是利用并行计算。通过保持一组规范参数，这些参数由多个网络副本异步读取和更新，计算可以有效地分布在单个中央处理单元（ CPU ）的两个处理核心上，以及跨集群机器中的多个 CPU。使用分布式系统，Nair 等人 [51] 开发了一个并行训练多个 DQN 的框架，实现了更好的性能和训练时间的减少。然而，为单机和分布式机器环境开发的更简单的异步优势 Actor-Critic （ A3C ）算法 [48],，近年来已成为最受欢迎的深度强化学习（ DRL ）技术之一。A3C 结合了优势更新和 Actor-Critic 公式，并依赖于在多个处理线程上并行训练的异步更新策略和价值函数网络。使用多个智能体，位于它们自己的独立环境中，不仅稳定了参数的改进，而且还带来了额外的优势，即允许进行更多探索。A3C 已被用作许多后续工作的标准起点，包括 Zhu 等人 [100], 将其应用于通过视觉输入在现实世界中进行的机器人导航的工作。

在原始 A3C 算法上已经取得了一些重大进展，这些进展反映了深度强化学习（ DRL ）领域的各种动机。首先是带有经验回放的 Actor-Critic [93], ，它为 A3C 添加了离线策略偏差校正，使其能够利用经验回放来提高样本复杂度。其他人则试图弥合基于值和基于策略的强化学习之间的差距，利用理论进展来改进原始的 A3C [50],[54]。最后，利用辅助任务来改进 DRL 代理学习到的表示的趋势正在增长，从而提高这些代理的学习速度和最终性能 [26], [46]。

当前研究及挑战

总之，我们将重点介绍 DRL 的当前研究领域以及仍然存在的挑战。之前，我们主要关注无模型方法，但现在我们将更详细地研究几种基于模型的 DRL 算法。基于模型的强化学习算法在提高强化学习数据效率和权衡探索与利用方面发挥着重要作用。在解决探索策略之后，我们将接着讨论分层

RL (HRL), which imposes an inductive bias on the final policy by explicitly factorizing it into several levels. When available, trajectories from other controllers can be used to bootstrap the learning process, leading us to imitation learning and inverse RL (IRL). For the final topic, we will look at multiagent systems, which have their own special considerations.

Model-based RL

The key idea behind model-based RL is to learn a transition model that allows for simulation of the environment without interacting with the environment directly. Model-based RL does not assume specific prior knowledge. However, in practice, we can incorporate prior knowledge (e.g., physics-based models [29]) to speed up learning. Model learning plays an important role in reducing the number of required interactions with the (real) environment, which may be limited in practice. For example, it is unrealistic to perform millions of experiments with a robot in a reasonable amount of time and without significant hardware wear and tear. There are various approaches to learn predictive models of dynamical systems using pixel information. Based on the deep dynamical model [90], where high-dimensional observations are embedded into a lower-dimensional space using autoencoders, several model-based DRL algorithms have been proposed for learning models and policies from pixel information [55], [91], [95]. If a sufficiently accurate model of the environment can be learned, then even simple controllers can be used to control a robot directly from camera images [14]. Learned models can also be used to guide exploration purely based on simulation of the environment, with deep models allowing these techniques to be scaled up to high-dimensional visual domains [75].

Although deep neural networks can make reasonable predictions in simulated environments over hundreds of time steps [10], they typically require many samples to tune the large number of parameters they contain. Training these models often requires more samples (interaction with the environment) than simpler models. For this reason, Gu et al. [19] train locally linear models for use with the NAF algorithm—the continuous equivalent of the DQN [47]—to improve the algorithm's sample complexity in the robotic domain where samples are expensive. It seems likely that the usage of deep models in model-based DRL could be massively spurred by general advances in improving the data efficiency of neural networks.

Exploration versus exploitation

One of the greatest difficulties in RL is the fundamental dilemma of exploration versus exploitation: When should the agent try out (perceived) nonoptimal actions to explore the environment (and potentially improve the model), and when should it exploit the optimal action to make useful progress? Off-policy algorithms, such as the DQN [47], typically use the simple ϵ-greedy exploration policy, which chooses a random action with probability $\epsilon \in [0, 1]$, and the optimal action otherwise. By decreasing ϵ over time, the agent progresses toward exploitation. Although adding independent noise for exploration is usable in continuous control problems, more sophisticated strategies inject noise that is correlated over time (e.g., from stochastic processes) to better preserve momentum [44].

The observation that temporal correlation is important led Osband et al. [56] to propose the bootstrapped DQN, which maintains several Q-value "heads" that learn different values through a combination of different weight initializations and bootstrapped sampling from experience replay memory. At the beginning of each training episode, a different head is chosen, leading to temporally extended exploration. Usunier et al. [85] later proposed a similar method that performed exploration in policy space by adding noise to a single output head, using zero-order gradient estimates to allow backpropagation through the policy.

One of the main principled exploration strategies is the upper confidence bound (UCB) algorithm, based on the principle of "optimism in the face of uncertainty" [36]. The idea behind UCB is to pick actions that maximize $\mathbb{E}[\mathbb{R}] + \kappa\sigma[R]$, where $\sigma[R]$ is the standard deviation of the return and $\kappa > 0$. UCB therefore encourages exploration in regions with high uncertainty and moderate expected return. While easily achievable in small tabular cases, the use of powerful density models has allowed this algorithm to scale to high-dimensional visual domains with DRL [4].

UCB can also be considered one way of implementing intrinsic motivation, which is a general concept that advocates decreasing uncertainty/making progress in learning about the environment [68]. There have been several DRL algorithms that try to implement intrinsic motivation via minimizing model prediction error [57], [75] or maximizing information gain [25], [49].

Hierarchical RL

In the same way that deep learning relies on hierarchies of features, HRL relies on hierarchies of policies. Early work in this area introduced options, in which, apart from primitive actions (single time-step actions), policies could also run other policies (multitime-step "actions") [79]. This approach allows top-level policies to focus on higher-level goals, while subpolicies are responsible for fine control. Several works in DRL have attempted HRL by using one top-level policy that chooses between subpolicies, where the division of states or goals in to subpolicies is achieved either manually [1], [34], [82] or automatically [2], [88], [89]. One way to help construct subpolicies is to focus on discovering and reaching goals, which are specific states in the environment; they may often be locations, to which an agent should navigate. Whether utilized with HRL or not, the discovery and generalization of goals is also an important area of ongoing research [35], [66], [89].

Imitation learning and inverse RL

One may ask why, if given a sequence of "optimal" actions from expert demonstrations, it is not possible to use supervised learning in a straightforward manner—a case of "learning from demonstration." This is indeed possible and is known as *behavioral cloning* in traditional RL literature. Taking advantage of the stronger signals available in supervised learning problems, behavioral cloning enjoyed success in earlier

RL （HRL），通过显式地将最终策略分解为多个级别，对最终策略施加归纳偏差。当可用时，可以从其他控制器中使用轨迹来启动学习过程，这使我们走向模仿学习和逆强化学习（IRL）。对于最后一个主题，我们将查看多智能体系统，它们有自己的特殊考虑。

基于模型的 *RL*

基于模型的 RL 背后的关键思想是学习一个转换模型，该模型允许在不直接与环境交互的情况下模拟环境。基于模型的 RL 不假设特定的先验知识。然而，在实践中，我们可以结合先验知识（例如，基于物理的模型 [29]）来加速学习。模型学习在减少与（真实）环境交互次数方面发挥着重要作用，这在实践中可能是有限的。例如，在合理的时间内用机器人进行数百万次实验是不现实的，而且没有显著的硬件磨损。有各种方法使用像素信息来学习动力系统的预测模型。基于深度动态模型 [90]，，其中使用自动编码器将高维观察嵌入到低维空间中，已经提出了几种基于模型的深度强化学习算法，用于从像素信息 [55], [91], [95] 中学习模型和政策。如果可以学习到足够精确的环境模型，那么即使使用简单的控制器，也可以直接从摄像头图像 [14] 控制机器人。学习的模型还可以用于纯粹基于模拟环境来指导探索，深度模型允许这些技术扩展到高维视觉领域 [75]。

尽管深度神经网络可以在模拟环境中对数百个时间步做出合理的预测 [10]，，但它们通常需要大量样本来调整其包含的大量参数。训练这些模型通常需要比简单模型更多的样本（与环境的交互）。因此，Gu 等人 [19] 训练局部线性模型以用于 NAF 算法 ——DQN[47] 的连续等价物 —— 以提高该算法在样本昂贵的机器人领域的样本复杂度。深度模型在基于模型的 DRL 中的使用似乎很可能被神经网络的通用数据效率提升所大量推动。

Exploration versus exploitation

强化学习中的一个最大困难是探索与利用的基本困境：当智能体应该尝试（感知到的）非最优动作来探索环境（并可能改进模型）时，以及何时应该利用最优动作以取得有用进展？离线算法，如 DQN [47]，通常使用简单的 ϵ- 贪婪探索策略，该策略以概率 $\in\epsilon[0, 1]$，选择随机动作，否则选择最优动作。通过随时间减少 ϵ，智能体将朝着利用方向进展。尽管在连续控制问题中添加独立的噪声用于探索是可行的，但更复杂的策略会注入与随时间变化（例如，来自随机过程）以更好地保持动量 [44]。

观察到时间相关性很重要，Osband 等人 [56] 提出了引导式 DQN，该算法维护了多个 Q- 值 " 头 "，通过不同的权重初始化和从经验回放内存中进行引导式采样来学习不同的值。在每个训练周期的开始，会选择一个不同的头，从而导致时间扩展的探索。Usunier 等人 [85] 后来提出了一种类似的方法，通过向单个输出头添加噪声来在策略空间中进行探索，使用零阶梯度估计来允许策略的反向传播。

一种主要的探索策略是上置信界 (UCB) 算法，基于 " 面对不确定性时的乐观 " 原则 [36]。UCB 的思想是选择最大化 $\mathbb{R}\mathbb{E}\ \mathbb{R}\ \kappa\sigma[\quad + \quad[]\,]$ 其中 $[\ ]R\sigma$ 是回报的标准差，。$0>\kappa$UCB 因此鼓励在具有高不确定性和适度预期回报的区域进行探索。虽然在小表格情况下很容易实现，但使用强大的密度模型使该算法能够扩展到具有深度强化学习 (DRL) [4] 的高维视觉领域。

UCB 也可以被视为一种实现内在动机的方式，内在动机是一个提倡减少不确定性 / 在学习环境中取得进展的一般概念 [68]。已经有一些深度强化学习算法尝试通过最小化模型预测误差 [57], [75] 或最大化信息增益 [25], [49] 来实现内在动机。

分层强化学习

与深度学习依赖于特征层次结构一样，分层强化学习也依赖于策略层次结构。该领域早期工作引入了选项，其中除了原始动作（单时间步动作）之外，策略还可以运行其他策略（多时间步 " 动作 "） [79]。这种方法允许顶层策略专注于高级目标，而子策略负责精细控制。深度强化学习中的几项工作尝试通过使用一个顶层策略在子策略之间进行选择来实现分层强化学习，其中状态或目标的子策略划分是通过手动 [1], [34], [82] 或自动 [2], [88], [89] 实现的。帮助构建子策略的一种方法是通过关注发现和达到目标，这些是环境中特定的状态；它们通常是位置，智能体应该导航到这些位置。无论是否与分层强化学习一起使用，目标的发现和泛化也是一个重要的持续研究领域 [35], [66], [89]。

模仿学习与逆强化学习

有人可能会问，如果从专家演示中给出一系列 " 最优 " 动作，为什么不能以直截了当的方式使用监督学习 —— 即 " 从演示中学习 "。这确实是可以做到的，在传统的强化学习文献中被称为行为克隆。利用监督学习问题中可用的更强信号，行为克隆在早期取得了成功

neural network research, with the most notable success being ALVINN, one of the earliest autonomous cars [59]. However, behavioral cloning cannot adapt to new situations, and small deviations from the demonstration during the execution of the learned policy can compound and lead to scenarios where the policy is unable to recover. A more generalizable solution is to use provided trajectories to guide the learning of suitable state-action pairs but fine-tune the agent using RL [23].

The goal of IRL is to estimate an unknown reward function from observed trajectories that characterize a desired solution [52]; IRL can be used in combination with RL to improve upon demonstrated behavior. Using the power of deep neural networks, it is now possible to learn complex, nonlinear reward functions for IRL [98]. Ho and Ermon [24] showed that policies are uniquely characterized by their occupancies (visited state and action distributions) allowing IRL to be reduced to the problem of measure matching. With this insight, they were able to use generative adversarial training [18] to facilitate reward-function learning in a more flexible manner, resulting in the generative adversarial imitation learning algorithm.

Multiagent RL

Usually, RL considers a single learning agent in a stationary environment. In contrast, multiagent RL (MARL) considers multiple agents learning through RL and often the nonstationarity introduced by other agents changing their behaviors as they learn [8]. In DRL, the focus has been on enabling (differentiable) communication between agents, which allows them to cooperate. Several approaches have been proposed for this purpose, including passing messages to agents sequentially [15], using a bidirectional channel (providing ordering with less signal loss) [58], and an all-to-all channel [77]. The addition of communication channels is a natural strategy to apply to MARL in complex scenarios and does not preclude the usual practice of modeling cooperative or competing agents as applied elsewhere in the MARL literature [8].

Conclusion: Beyond pattern recognition

Despite the successes of DRL, many problems need to be addressed before these techniques can be applied to a wide range of complex real-world problems [37]. Recent work with (nondeep) generative causal models demonstrated superior generalization over standard DRL algorithms [48], [63] in some benchmarks [5], achieved by reasoning about causes and effects in the environment [29]. For example, the schema networks of Kanksy et al. [29] trained on the game *Breakout* immediately adapted to a variant where a small wall was placed in front of the target blocks, while progressive (A3C) networks [63] failed to match the performance of the schema networks even after training on the new domain. Although DRL has already been combined with AI techniques, such as search [73] and planning [80], a deeper integration with other traditional AI approaches promises benefits such as better sample complexity, generalization, and interpretability [16]. In time, we also hope that our theoretical understanding of the properties of neural networks (particularly within DRL) will improve, as it currently lags far behind practice.

To conclude, it is worth revisiting the overarching goal of all of this research: the creation of general-purpose AI systems that can interact with and learn from the world around them. Interaction with the environment is simultaneously the advantage and disadvantage of RL. While there are many challenges in seeking to understand our complex and ever-changing world, RL allows us to choose how we explore it. In effect, RL endows agents with the ability to perform experiments to better understand their surroundings, enabling them to learn even high-level causal relationships. The availability of high-quality visual renderers and physics engines now enables us to take steps in this direction, with works that try to learn intuitive models of physics in visual environments [13]. Challenges remain before this will be possible in the real world, but steady progress is being made in agents that learn the fundamental principles of the world through observation and action. Perhaps, then, we are not too far away from AI systems that learn and act in more human-like ways in increasingly complex environments.

> **In effect, RL endows agents with the ability to perform experiments to better understand their surroundings, enabling them to learn even high-level causal relationships.**

Acknowledgments

Kai Arulkumaran would like to acknowledge Ph.D. funding from the Department of Bioengineering at Imperial College London. This research has been partially funded by a Google Faculty Research Award to Marc Deisenroth.

Authors

Kai Arulkumaran (ka709@imperial.ac.uk) received a B.A. degree in computer science at the University of Cambridge in 2012 and an M.Sc. degree in biomedical engineering from Imperial College London in 2014, where he is currently a Ph.D. degree candidate in the Department of Bioengineering. He was a research intern at Twitter's Magic Pony and Microsoft Research in 2017. His research focus is deep reinforcement learning and transfer learning for visuomotor control.

Marc Peter Deisenroth (m.deisenroth@imperial.ac.uk) received an M.Eng. degree in computer science at the University of Karlsruhe in 2006 and a Ph.D. degree in machine learning at the Karlsruhe Institute of Technology in 2009. He is a lecturer of statistical machine learning in the Department of Computing at Imperial College London and with PROWLER.io. He was awarded an Imperial College Research Fellowship in 2014 and received Best Paper Awards at the International Conference on Robotics and Automation 2014 and the International Conference on Control, Automation, and Systems 2016. He is a recipient of a Google Faculty Research Award and a Microsoft Ph.D. Scholarship. His research is centered around data-efficient machine learning for autonomous decision making.

Miles Brundage (miles.brundage@philosophy.ox.ac.uk) received a B.A. degree in political science at George

神经网络研究，最显著的成就是早期自动驾驶汽车之一 ALVINN [59]。然而，行为克隆无法适应新情况，执行学习到的策略时演示中的微小偏差会累积，导致策略无法恢复。更通用的解决方案是使用提供的轨迹来指导合适的状态 - 动作对的学习，但使用强化学习 [23] 对智能体进行微调。

The goal of IRL is to estimate an unknown reward function from observed trajectories that characterize a desired solution [52]; IRL 可以与 RL 结合使用，以改进展示的行为。利用深度神经网络的强大功能，现在可以学习用于 IRL 的复杂非线性奖励函数 [98]。Ho 和 Ermon [24] 证明了策略由其占用率（访问状态和动作分布）唯一表征，这使得 IRL 可以简化为度量匹配问题。基于这一见解，他们能够使用生成对抗训练 [18] 来促进
以更灵活的方式学习奖励函数，从而产生了生成对抗性模仿学习算法。

> 实际上，强化学习赋予智能体进行实验的能力，以更好地理解其周围环境，使他们能够学习高级因果关系。

多智能体强化学习

通常，强化学习考虑在静态环境中运行的单个学习智能体。相比之下，多智能体强化学习（MARL）考虑多个智能体通过强化学习进行学习，并且通常其他智能体在学习过程中改变其行为引入的非平稳性 [8]。在深度强化学习中，重点在于实现（可微分的）智能体之间的通信，这允许它们合作。为了这个目的，已经提出了几种方法，包括按顺序向智能体传递消息 [15], , 使用双向通道（提供顺序且信号损失较少）[58], 和全对全通道 [77]。添加通信通道是应用于复杂场景 MARL 的一种自然策略，并且不排除在 MARL 文献的其他地方建模合作或竞争智能体的通常做法 [8]。

结论：超越模式识别

尽管深度强化学习取得了成功，但在将这些技术应用于广泛的复杂现实世界问题 [37] 之前，还有许多问题需要解决。最近使用（非深度）生成因果模型的工作在一些基准测试 [5], 中展示了优于标准深度强化学习算法 [48], [63] 的泛化能力 [29]，这是通过推理环境中的因果关系实现的 [29] 。例如，Kanksy 等人 [29] 在游戏 <code id='6'>Break- out</code> 上训练的架构网络立即适应了在目标块前放置一个小墙的变体，而渐进式（A3C）网络 [63] 即使在新的领域上进行训练后，也无法匹配架构网络的性能。尽管深度强化学习已经与搜索 [73] 和规划 [80], 等人工智能技术相结合，但与其他传统人工智能方法的更深入集成有望带来诸如更好的好处等。

样本复杂度、泛化能力和可解释性 [16]。随着时间的推移，我们也希望我们的神经网络（尤其是深度强化学习）的性质的理论理解能够得到提高，因为它目前远远落后于实践。

总结来说，值得重新审视这项研究的总体目标：创建能够与周围世界互动并从中学习的通用人工智能系统。与环境的互动既是强化学习的优势也是劣势。虽然理解我们复杂且不断变化的世界存在许多挑战，但强化学习允许
我们选择如何探索它。实际上，强化学习赋予智能体进行实验以更好地理解周围环境的能力，使他们能够学习甚至高级的因果关系。现在高质量的可视渲染器和物理引擎的可用性使我们能够朝着这个方向发展，有作品试图在视觉环境中学习直观的物理模型 [13]。在现实世界中实现这一点之前仍然存在挑战，但稳步的进展
正在智能体中实现，这些智能体通过观察和行动学习世界的根本原理。也许，那时我们距离能够在日益复杂的环境中更类似人类地学习和行动的 AI 系统已经不远了。

Acknowledgments

Kai Arulkumaran 希望感谢伦敦帝国理工学院生物工程系的博士资助。这项研究部分得到了谷歌教职研究奖的资助，获奖者为 Marc Deisenroth。

作者

Kai Arulkumaran (ka709@imperial.ac.uk) 于 2012 年在剑桥大学获得计算机科学学士学位，并于 2014 年在伦敦帝国理工学院获得生物医学工程硕士学位，目前在该校生物工程系攻读博士学位。他在 2017 年是 Twitter 的 Magic Pony 和微软研究机构的实习研究员。他的研究重点是深度强化学习和视觉运动控制的迁移学习。

Marc Peter Deisenroth (m.deisenroth@imperial.ac.uk) 于 2006 年在卡尔斯鲁厄大学获得计算机科学硕士学位，并于 2009 年在卡尔斯鲁厄理工学院获得机器学习博士学位。他是伦敦帝国学院计算系和 PROWLER.io 的统计机器学习讲师。他在 2014 年获得帝国学院研究奖学金，并在 2014 年国际机器人与自动化会议和 2016 年国际控制、自动化与系统会议上获得最佳论文奖。他是谷歌教师研究奖和微软博士奖学金的获得者。他的研究主要集中在数据高效的机器学习，用于自主决策。

Miles Brundage (miles.brundage@philosophy.ox.ac.uk) 在政治学专业获得文学士学位，就读于乔治

Washington University, Washington, D.C., in 2010. He is a Ph.D. degree candidate in the Human and Social Dimensions of Science and Technology Department at Arizona State University and a research fellow at the University of Oxford's Future of Humanity Institute. His research focuses on governance issues related to artificial intelligence.

Anil Anthony Bharath (a.bharath@ic.ac.uk) received his B. Eng. degree in electronic and electrical engineering from University College London in 1988 and his Ph.D. degree in signal processing from Imperial College London in 1993, where he is currently a reader in the Department of Bioengineering. He is also a fellow of the Institution of Engineering and Technology and a cofounder of Cortexica Vision Systems. He was previously an academic visitor in the Signal Processing Group at the University of Cambridge in 2006. His research interests are in deep architectures for visual inference.

References

[1] K. Arulkumaran, N. Dilokthanakul, M. Shanahan, and A. A. Bharath, "Classifying options for deep reinforcement learning," in *Proc. IJCAI Workshop Deep Reinforcement Learning: Frontiers and Challenges*, 2016.

[2] P. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proc. Association Advancement Artificial Intelligence*, 2017, pp. 1726–1734.

[3] L. C. Baird III, "Advantage updating," Defense Tech. Inform. Center, Tech. Report D-A280 862, Fort Belvoir, VA, 1993.

[4] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," in *Proc. Neural Information Processing Systems*, 2016, pp. 1471–1479.

[5] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: an evaluation platform for general agents," in *Proc. Int. Joint Conf. Artificial Intelligence*, 2015, pp. 253–279.

[6] R. Bellman, "On the theory of dynamic programming," *Proc. Nat. Acad. Sci*., vol. 38, no. 8, pp. 716–719, 1952.

[7] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell*., vol. 35, no. 8, pp. 1798–1828, 2013.

[8] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern.*, vol. 38, no. 2, pp. 156–172, 2008.

[9] M. Campbell, A. J. Hoane, and F. Hsu, "Deep Blue," *Artificial Intell*., vol. 134, no. 1-2, pp. 57–83, 2002.

[10] S. Chiappa, S. Racaniere, D. Wierstra, and S. Mohamed, "Recurrent environment simulators," in *Proc. Int. Conf. Learning Representations*, 2017.

[11] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba. (2016). Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv*. [Online]. Available: https://arxiv.org/abs/1610.03518

[12] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[13] M. Denil, P. Agrawal, T. D. Kulkarni, T. Erez, P. Battaglia, and N. de Freitas, "Learning to perform physics experiments via deep reinforcement learning," in *Proc. Int. Conf. Learning Representations*, 2017.

[14] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2016, pp. 512–519.

[15] J. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Neural Information Processing Systems*, 2016, pp. 2137–2145.

[16] M. Garnelo, K. Arulkumaran, and M. Shanahan, "Towards deep symbolic reinforcement learning," in *NIPS Workshop on Deep Reinforcement Learning*, 2016.

[17] F. Gomez and J. Schmidhuber. "Evolving modular fast-weight networks for control," in *Proc. Int. Conf. Artificial Neural Networks*, 2005, pp. 383–389.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Neural Information Processing Systems*, 2014, pp. 2672–2680.

[19] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," in *Proc. Int. Conf. Learning Representations*, 2016.

[20] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," in *Association for the Advancement of Artificial Intelligence Fall Symp. Series*, 2015.

[21] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver. "Memory-based control with recurrent neural networks," in *NIPS Workshop on Deep Reinforcement Learning*, 2015.

[22] N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa, "Learning continuous control policies by stochastic value gradients," in *Proc. Neural Information Processing Systems*, 2015, pp. 2944–2952.

[23] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, A. Sendonaris, G. Dulac-Arnold, et al. (2017). Learning from demonstrations for real world reinforcement learning. *arXiv*. [Online]. Available: https://arxiv.org/abs/1704.03732

[24] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Neural Information Processing Systems*, 2016, pp. 4565-4573.

[25] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. de Turck, and P. Abbeel, "VIME: Variational information maximizing exploration," in *Proc. Neural Information Processing Systems*, 2016, pp. 1109–1117.

[26] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," in *Proc. Int. Conf. Learning Representations*, 2017.

[27] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intell*., vol. 101, no. 1, pp. 99–134, 1998.

[28] S. M. Kakade, "A natural policy gradient," in *Proc. Neural Information Processing Systems*, 2002, pp. 1531–1538.

[29] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, S. Phoenix, and D. George, "Schema networks: zero-shot transfer with a generative causal model of intuitive physics," in *Proc. Int. Conf. Machine Learning*, 2017, pp. 1809–1818.

[30] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. Int. Conf. Learning Representations*, 2014.

[31] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2004, pp. 2619–2624.

[32] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM J. Control Optim.*, vol. 42, no. 4, pp. 1143–1166, 2003.

[33] J. Koutník, G. Cuccu, J. Schmidhuber, and F. Gomez, "Evolving large-scale neural networks for vision-based reinforcement learning," in *Proc. Conf. Genetic and Evolutionary Computation*, 2013, pp. 1061–1068.

[34] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Neural Information Processing Systems*, 2016, pp. 3675–3683.

[35] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman, "Deep successor reinforcement learning," in *NIPS Workshop on Deep Reinforcement Learning*, 2016.

[36] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. Appl. Math*., vol. 6, no. 1, pp. 4–22, 1985.

[37] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral Brain Sci*., pp. 1–101, 2016. [Online]. Available: https://www.cambridge.org/core/journals/behavioral-and-brain-sciences/article/building-machines-that-learn-and-think-like-people/A9535B1D745A0377E16C590E14B94993

[38] S. Lange, M. Riedmiller, and A. Voigtlander, "Autonomous reinforcement learning on raw visual input data in a real world application," in *Proc. Int. Joint Conf. Neural Networks*, 2012, pp. 1–8.

[39] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[40] S. Levine and V. Koltun, "Guided policy search," in *Proc. Int. Conf. Learning Representations*, 2013.

[41] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learning Res*., vol. 17, no. 39, pp. 1–40, 2016.

[42] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," in *Proc. Int. Symp. Experimental Robotics*, 2016, pp. 173–184.

[43] Y. Li. (2017). Deep reinforcement learning: An overview. *arXiv*. [Online]. Available: https://arxiv.org/abs/1701.07274

[44] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learning Representations*, 2016.

[45] L. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learning*, vol. 8, no. 3–4, pp. 293–321, 1992.

[46] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, et al., "Learning to navigate in complex environments," in *Proc. Int. Conf. Learning Representations*, 2017.

[47] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

华盛顿大学，华盛顿特区，2010 年。他是亚利桑那州立大学科学技术人类与社会维度系的博士候选人，也是牛津大学未来人类研究所的研究员。他的研究重点是人工智能相关的治理问题。

Anil Anthony Bharath (a.bharath@ic.ac.uk) 于 1988 年在伦敦大学学院获得电子电气工程学士学位，并于 1993 年在伦敦帝国学院获得信号处理博士学位，目前在该学院的生物工程系担任研究员。他还是英国工程师学会的会员，也是 Cortexica Vision Systems 的联合创始人。他曾在 2006 年作为访问学者在剑桥大学的信号处理小组工作。他的研究兴趣在于用于视觉推理的深度架构。

参考文献

[1] K. Arulkumaran, N. Dilokthanakul, M. Shanahan, and A. A. Bharath, "深度强化学习的选项分类，" in *IJCAI* 工作坊深度强化学习：前沿与挑战 2016.

[2] P. Bacon, J. Harb, and D. Precup, "选项 - 评价器架构，" in 人工智能协会进展 , 2017, pp. 1726–1734.

[3] L. C. Baird III, "优势更新，" 国防技术信息中心 , 技术报告 D-A280 862, 弗吉尼亚州贝尔沃 Fort, 1993.

[4] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "统一计数式探索和内在动机，" in 神经信息处理系统 , 2016, pp. 1471–1479.

[5] M. G. Bellemare, Y. Naddaf, J. Veness, 和 M. Bowling, "The arcade learning environment: an evaluation platform for general agents," 在 *Proc. Int. Joint Conf Artif icial Intelligence*, 2015, pp. 253–279.

[6] R. Bellman, "On the theory of dynamic programming," *Proc. Nat. Acad. Sci*., vol. 38, no. 8, pp. 716–719, 1952.

[7] Y. Bengio, A. Courville, 和 P. Vincent, "Representation learning: a review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell*., vol. 35, no. 8, pp. 1798–1828, 2013.

[8] L. Busoniu, R. Babuska, 和 B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern.*, vol. 38, no. 2, pp. 156–172, 2008.

[9] M. Campbell, A. J. Hoane, 和 F. Hsu, "Deep Blue," 人工智能 ., vol. 134, no. 1-2, pp. 57–83, 2002.

[10] S. Chiappa, S. Racaniere, D. Wierstra, 和 S. Mohamed, "循环环境模拟器，" 在 国际学习表示会议论文集 , 2017.

[11] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, 和 W. Zaremba. (2016). 通过学习深度逆动力模型从模拟到真实世界的迁移 . *arXiv*. [在线]. 可用 : https://arxiv.org/ abs/1610.03518

[12] M. P. Deisenroth, G. Neumann, 和 J. Peters, "机器人策略搜索综述，" 机器人基础与趋势 , vol. 2, no. 1–2, pp. 1–142, 2013.

[13] M. Denil, P. Agrawal, T. D. Kulkarni, T. Erez, P. Battaglia, and N. de Freitas, "通过深度强化学习学习执行物理实验，" in 国际学习表示会议论文集 , 2017.

[14] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "用于视觉运动学习的深空空间自动编码器，" in *IEEE* 国际机器人与自动化会议论文集 , 2016, pp. 512–519.

[15] J. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "通过深度多智能体强化学习学习交流，" in 神经信息处理系统会议论文集 , 2016, pp. 2137–2145.

[16] M. Garnelo, K. Arulkumaran, and M. Shanahan, "迈向深度符号强化学习，" in *NIPS* 深度强化学习研讨会 , 2016.

[17] F. Gomez and J. Schmidhuber. "进化模块化快速权重网络用于控制，" in *Proc. Int. Conf . Artif icial Neural Networks*, 2005, pp. 383–389.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "生成对抗网络，" in *Proc. Neural Inf ormationProcessing Systems*, 2014, pp. 2672–2680.

[19] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "基于模型的连续深度 Q 学习加速，" in *Proc. Int. Conf . Learning Representations*, 2016.

[20] M. Hausknecht and P. Stone, "深度循环 Q 学习用于部分可观察 MDPs," in *Association f or the Advancement of Artif icial Intelligence FallSymp. Series*, 2015.

[21] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver. "基于记忆的控制与循环神经网络，" 在 深度强化学习研讨会 , 2015.

[22] N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, 和 Y. Tassa, "通过随机值梯度学习连续控制策略，" 在 神经信息处理系统会议论文集 , 2015, pp. 2944–2952.

[23] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, A. Sendonaris, G. Dulac-Arnold, 等 . (2017). 基于演示的现实世界强化学习 . *arXiv*. [在线]. 可获得 : https://arxiv.org/ abs/1704.03732

[24] J. Ho 和 S. Ermon, "生成对抗性模仿学习，" 在《神经信息处理系统会议论文集》 , 2016, pp. 4565-4573.

[25] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. de Turck, 和 P. Abbeel, "VIME: 变分信息最大化探索，" 在《神经信息处理系统会议论文集》 , 2016, pp. 1109–1117.

[26] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, 和 K. Kavukcuoglu, "带无监督辅助任务的强化学习，" 在《国际学习表示会议论文集》 , 2017.

[27] L. P. Kaelbling, M. L. Littman, 和 A. R. Cassandra, "在部分可观察随机域中的规划和行动，" 《人工智能》 , 第 101 卷 , 第 1 期 , pp. 99–134, 1998.

[28] S. M. Kakade, "自然策略梯度，" 在 *Proc. Neural Inf ormationProcessing Systems*, 2002, pp. 1531–1538.

[29] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, S. Phoenix, 和 D. George, "模式网络：使用直观物理生成因果模型的零样本迁移，" 在 *Proc. Int. Conf . MachineLearning*, 2017, pp. 1809–1818.

[30] D. P. Kingma 和 M. Welling, "自编码变分贝叶斯，" 在 *Proc. Int.Conf . Learning Representations*, 2014.

[31] N. Kohl 和 P. Stone, "策略梯度强化学习用于快速四足运动，" 在 *Proc. IEEE Int. Conf . Robotics and Automation*, 2004, pp. 2619– 2624.

[32] V. R. Konda and J. N. Tsitsiklis, "基于演员 - 评论家算法的研究，" *SIAM J. ControlOptim.*, vol. 42, no. 4, pp. 1143–1166, 2003.

[33] J. Koutník, G. Cuccu, J. Schmidhuber, and F. Gomez, "为视觉强化学习进化大规模神经网络，" in 遗传与进化计算会议论文集 , 2013, pp. 1061–1068.

[34] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "分层深度强化学习：整合时间抽象和内在动机，" in 神经信息处理系统会议论文集 , 2016, pp. 3675–3683.

[35] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman, "深度后继强化学习，" in *NIPS* 深度强化学习研讨会 , 2016.

[36] T. L. Lai and H. Robbins, "渐近有效自适应分配规则，" *AdvAppl. Math.*, vol. 6, no. 1, pp. 4–22, 1985.

[37] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "构建像人类一样学习和思考的机器，" *Behavioral Brain Sci*., pp. 1–101, 2016.[在线]. 可用 : https://www.cambridge.org/core/journals/behavioral-and-brain-sciences/article/building-machines-that-learn-and-think-like-people/A9535B1D745A0377E16C590E14B94993

[38] S. Lange, M. Riedmiller, and A. Voigtlander, "在真实世界应用中基于原始视觉输入数据的自主强化学习，" in *Proc. Int. Joint Conf .Neural Networks*, 2012, pp. 1–8.

[39] Y. LeCun, Y. Bengio, 和 G. Hinton. "深度学习，" *Nature*, 第 521 卷 , 第 7553 期 , 第 436–444 页 , 2015.

[40] S. Levine 和 V. Koltun, "引导策略搜索，" 在 国际学习表征会议论文集 , 2013.

[41] S. Levine, C. Finn, T. Darrell, 和 P. Abbeel, "深度视觉运动策略的端到端训练，" 机器学习研究杂志 , 第 17 卷 , 第 39 期 , 第 1–40 页 , 2016.

[42] S. Levine, P. Pastor, A. Krizhevsky, 和 D. Quillen, "使用深度学习和大规模数据收集学习机械手 - 眼睛协调以进行机器人抓取，" 在 国际实验机器人会议论文集 , 2016, 第 173–184 页 .

[43] Y. Li. (2017). 深度强化学习：概述 . *arXiv*. [在线]. 可获得 : https://arxiv.org/abs/1701.07274

[44] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, 和 D. Wierstra, "深度强化学习的连续控制，" 在 国际会议学习表示过程 , 2016.

[45] L. Lin, "基于强化学习、规划和教学的自我改进反应代理，" 机器学习 , 第 8 卷 , 第 3-4 期 , 第 293-321 页 , 1992.

[46] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, 等 ., "在复杂环境中导航的学习，" 在 国际会议学习表示过程 , 2017.

[47] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, et al., "深度强化学习实现人类水平控制，" *Nature*, 第518 卷 , 第 7540 期 , 第 529–533 页 , 2015.

[48] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Learning Representations*, 2016.

[49] S. Mohamed and D. J. Rezende, "Variational information maximisation for intrinsically motivated reinforcement learning," in *Proc. Neural Information Processing Systems*, 2015, pp. 2125–2133.

[50] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans. (2017). Bridging the gap between value and policy based reinforcement learning. *arXiv*. [Online]. Available: https://arxiv.org/abs/1702.08892

[51] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. de Maria, V. Panneershelvam, M. Suleyman, et al., "Massively parallel methods for deep reinforcement learning," in *ICML Workshop on Deep Learning*, 2015.

[52] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proc. Int. Conf. Machine Learning*, 2000, pp. 663–670.

[53] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *Proc. Int. Symp. Experimental Robotics*, 2006, pp. 363–372.

[54] B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih, "PGQ: Combining policy gradient and Q-learning," in *Proc. Int. Conf. Learning Representations*, 2017.

[55] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in Atari games," in *Proc. Neural Information Processing Systems*, 2015, pp. 2863–2871.

[56] I. Osband, C. Blundell, A. Pritzel, and B. van Roy, "Deep exploration via bootstrapped DQN," in *Proc. Neural Information Processing Systems*, 2016, pp. 4026–4034.

[57] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. Int. Conf. Machine Learning*, 2017, pp. 2778–2787.

[58] P. Peng, Q. Yuan, Y. Wen, Y. Yang, Z. Tang, H. Long, and J. Wang. (2017). Multiagent bidirectionally-coordinated nets for learning to play StarCraft combat games. *arXiv*. [Online]. Available: https://arxiv.org/abs/1703.10069

[59] D. A. Pomerleau, "ALVINN, an autonomous land vehicle in a neural network," in *Proc. Neural Information Processing Systems*, 1989, pp. 305–313.

[60] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proc. Int. Conf. Machine Learning*, 2014, pp. 1278–1286.

[61] M. Riedmiller, "Neural fitted q iteration—First experiences with a data efficient neural reinforcement learning method," in *Proc. European Conf. Machine Learning*, 2005, pp. 317–328.

[62] G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," Dept. Engineering, Univ. Cambridge, MA, Tech. Rep. CUED/F-INFENG/TR 166, 1994.

[63] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. (2016). Progressive neural networks. *arXiv*. [Online]. Available: https://arxiv.org/abs/1606.04671

[64] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. (2016). Sim-to-real robot learning from pixels with progressive nets. *arXiv*. [Online]. Available: https://arxiv.org/abs/1610.04286

[65] T. Salimans, J. Ho, X. Chen, and I. Sutskever. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv*. [Online]. Available: https://arxiv.org/abs/1703.03864

[66] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proc. Int. Conf. Machine Learning*, 2015, pp. 1312–1320.

[67] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. Int. Conf. Learning Representations*, 2016.

[68] J. Schmidhuber, "A possibility for implementing curiosity and boredom in model-building neural controllers," in *Proc. Int. Conf. Simulation Adaptive Behavior*, 1991, pp. 222–227.

[69] J. Schmidhuber and R. Huber, "Learning to generate artificial fovea trajectories for target detection," *Int. J. Neural Syst.*, vol. 2, no. 01n02, pp. 125–134, 1991. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S012906579100011X

[70] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Machine Learning*, 2015, pp. 1889–1897.

[71] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. Int. Conf. Learning Representations*, 2016.

[72] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Machine Learning*, 2014, pp. 387–395.

[73] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[74] S. Singh, D. Litman, M. Kearns, and M. Walker, "Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system," *J. Artificial Intell. Res.*, vol. 16, pp. 105–133, Feb. 2002.

[75] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," in *NIPS Workshop on Deep Reinforcement Learning*, 2015.

[76] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, "PAC model-free reinforcement learning," in *Proc. Int. Conf. Machine Learning*, 2006, pp. 881–888.

[77] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Proc. Neural Information Processing Systems*, 2016, pp. 2244–2252.

[78] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[79] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial Intell.*, vol. 112, no. 1–2, pp. 181–211, 1999.

[80] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," in *Proc. Neural Information Processing Systems*, 2016, pp. 2154–2162.

[81] G. Tesauro, "Temporal difference learning and TD-gammon," *Commun. ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[82] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor, "A deep hierarchical approach to lifelong learning in Minecraft," in *Proc. Association for the Advancement Artificial Intelligence*, 2017, pp. 1553–1561.

[83] J. N. Tsitsiklis and B. van Roy, "Analysis of temporal-difference learning with function approximation," in *Proc. Neural Information Processing Systems*, 1997, pp. 1075–1081.

[84] E. Tzeng, C. Devin, J. Hoffman, C. Finn, X. Peng, S. Levine, K. Saenko, and T. Darrell, "Towards adapting deep visuomotor representations from simulated to real environments," in *Workshop Algorithmic Foundations Robotics*, 2016.

[85] N. Usunier, G. Synnaeve, Z. Lin, and S. Chintala, "Episodic exploration for deep deterministic policies: An application to StarCraft micromanagement tasks," in *Proc. Int. Conf. Learning Representations*, 2017.

[86] H. van Hasselt, "Double Q-learning," in *Proc. Neural Information Processing Systems*, 2010, pp. 2613–2621.

[87] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. Association for the Advancement of Artificial Intelligence*, 2016, pp. 2094–2100.

[88] A. Vezhnevets, V. Mnih, S. Osindero, A. Graves, O. Vinyals, J. Agapiou, and K. Kavukcuoglu. "Strategic attentive writer for learning macro-actions," in *Proc. Neural Information Processing Systems*, 2016, pp. 3486–3494.

[89] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "FeUdal networks for hierarchical reinforcement learning," in *Proc. Int. Conf. Machine Learning*, 2017, pp. 3540–3549.

[90] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "Learning deep dynamical models from image pixels," in *Proc. IFAC Symp. System Identification*, 2015, pp. 1059–1064.

[91] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "From pixels to torques: policy learning with deep dynamical models," in *ICML Workshop on Deep Learning*, 2015.

[92] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Learning Representations*, 2016.

[93] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," in *Proc. Int. Conf. Learning Representations*, 2017.

[94] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[95] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Proc. Neural Information Processing Systems*, 2015, pp. 2746–2754.

[96] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber, "Recurrent policy gradients," *Logic J. IGPL*, vol. 18, no. 5, pp. 620–634, 2010.

[97] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learning*, vol. 8, no. 3–4, pp. 229–256, 1992.

[98] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," in *NIPS Workshop on Deep Reinforcement Learning*, 2015.

[99] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. Machine Learning*, 2015, pp. 2048–2057.

[100] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2017, pp. 3357–3364.

[48] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "异步深度强化学习方法," 在 国际学习表征会议论文集 , 2016.

[49] S. Mohamed and D. J. Rezende, "内在动机强化学习的变分信息最大化," 在 神经信息处理系统会议论文集 , 2015, pp. 2125–2133.

[50] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans. (2017). 桥接基于值和基于策略的强化学习之间的差距 . *arXiv*. [在线]. 可用 : https://arxiv.org/abs/1702.08892

[51] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. de Maria, V. Panneershelvam, M. Suleyman, et al., "大规模并行深度强化学习方法," in *ICML Workshop on Deep Learning*, 2015.

[52] A. Y. Ng and S. J. Russell, "逆强化学习算法," in *Proc. Int. Conf . Machine Learning*, 2000, pp. 663–670.

[53] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "通过强化学习实现自主倒飞直升机," in *Proc. Int. Symp. Experimental Robotics*, 2006, pp. 363–372.

[54] B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih, "PGQ: 结合策略梯度和 Q 学习," in *Proc. Int. Conf . Learning Representations*, 2017.

[55] J. Oh, X. Guo, H. Lee, R. L. Lewis, 和 S. Singh, "使用深度网络在 Atari 游戏中进行动作条件视频预测," 在 神经信息处理系统会议论文集 , 2015, pp. 2863–2871.

[56] I. Osband, C. Blundell, A. Pritzel, 和 B. van Roy, "通过引导的 DQN 进行深度探索," 在 神经信息处理系统会议论文集 , 2016, pp. 4026– 4034.

[57] D. Pathak, P. Agrawal, A. A. Efros, 和 T. Darrell, "通过自监督预测进行好奇心驱动的探索," 在 国际机器学习会议论文集 , 2017, pp. 2778–2787.

[58] P. Peng, Q. Yuan, Y. Wen, Y. Yang, Z. Tang, H. Long, 和 J. Wang. (2017). 用于学习玩星际争霸战斗游戏的多人智能体双向协调网络 . *arXiv*. [在线]. 可用 : https://arxiv.org/abs/1703.10069

[59] D. A. Pomerleau, "ALVINN, 一个在神经网络中的自主陆地车辆," in *Proc. Neural Inf ormation Processing Systems*, 1989, pp. 305–313.

[60] D. J. Rezende, S. Mohamed, and D. Wierstra, "随机反向传播和深度生成模型中的近似推理," in *Proc. Int. Conf . MachineLearning*, 2014, pp. 1278–1286.

[61] M. Riedmiller, "神经拟合 q 迭代 —— 一种数据高效的神经强化学习方法的第一经验," in *Proc. European Conf . MachineLearning*, 2005, pp. 317–328.

[62] G. A. Rummery and M. Niranjan, "使用连接系统进行在线 Q 学习," Dept. Engineering, Univ. Cambridge, MA, Tech. Rep. CUED/F-INFENG/TR 166, 1994.

[63] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. (2016). Progressive neural networks. *arXiv*. [Online]. Available: https://arxiv.org/abs/1606.04671

[64] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. (2016). Sim-to-real robot learning from pixels with progressive nets. *arXiv*. [Online]. Available: https://arxiv.org/abs/1610.04286

[65] T. Salimans, J. Ho, X. Chen, and I. Sutskever. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv*. [Online]. Available: https://arxiv.org/abs/1703.03864

[66] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "通用价值函数逼近器," 在 国际机器学习会议论文集 , 2015, pp. 1312–1320.

[67] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "优先经验回放," 在 学习表示国际会议论文集 , 2016.

[68] J. Schmidhuber, "一种在建模神经控制器中实现好奇心和无聊的可能性," 在 模拟自适应行为国际会议论文集 , 1991, pp. 222–227.

[69] J. Schmidhuber and R. Huber, "学习生成用于目标检测的人工视锥轨迹," 国际神经系统杂志 , vol. 2, no. 01n02, pp. 125–134, 1991.[在线]. 可用 : http://www.worldscientific.com/doi/abs/10.1142/ S012906579100011X

[70] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "信任区域策略优化," 在 国际机器学习会议论文集 , 2015, pp. 1889–1897.

[71] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "高维连续控制使用广义优势估计," 在 学习表示会议论文集 , 2016.

[72] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "确定性策略梯度算法," 在 国际机器学习会议论文集 , 2014, pp. 387–395.

[73] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, 等 ., "使用深度神经网络和树搜索掌握围棋," 自然 , 第 529 卷 , 第 7587 期 , pp. 484–489, 2016.

[74] S. Singh, D. Litman, M. Kearns, and M. Walker, "使用强化学习优化对话管理：NJFun 系统的实验," *J.* 人工智能研究 , 第 16 卷 , 第 105-133 页 , 2002 年 2 月 .

[75] B. C. Stadie, S. Levine, and P. Abbeel, "使用深度预测模型在强化学习中激励探索," 在 *NIPS* 深度强化学习研讨会 , 2015 年 .

[76] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, "PAC 无模型强化学习," 在 国际机器学习会议论文集 , 2006 年 , 第 881-888 页 .

[77] S. Sukhbaatar, A. Szlam, and R. Fergus, "使用反向传播学习多智能体通信," 在 神经信息处理系统会议论文集 , 2016 年 , 第 2244-2252 页 .

[78] R. S. Sutton and A. G. Barto, 强化学习：一种介绍 . 马萨诸塞州剑桥：麻省理工学院出版社，1998.

[79] R. S. Sutton, D. Precup, and S. Singh, "介于 MDPs 和半 -MDPs 之间：强化学习中时间抽象的框架," 人工智能 , 第 112 卷 , 第 1-2 期 , 第 181-211 页 , 1999.

[80] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "值迭代网络," 在 神经信息处理系统会议论文集 , 2016, 第 2154-2162 页 .

[81] G. Tesauro, "时序差分学习和 TD-gammon," *ACM* 通讯 , 第 38 卷 , 第 3 期 , 第 58-68 页 , 1995.

[82] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor, "一种深度层次方法用于 Minecraft 中的终身学习," 在 人工智能进步协会会议论文集 , 2017, 第 1553-1561 页 .

[83] J. N. Tsitsiklis 和 B. van Roy, "具有函数近似的时序差分学习分析," 在 神经信息处理系统会议论文集 , 1997, pp. 1075–1081.

[84] E. Tzeng, C. Devin, J. Hoffman, C. Finn, X. Peng, S. Levine, K. Saenko, 和 T. Darrell, "从模拟环境到真实环境适应深度视觉运动表示," 在 机器人算法基础研讨会 , 2016.

[85] N. Usunier, G. Synnaeve, Z. Lin, 和 S. Chintala, "用于深度确定性策略的情景探索：应用于星际争霸微观管理任务," 在 国际学习表示会议论文集 , 2017.

[86] H. van Hasselt, "双 Q 学习," 在 神经信息处理系统会议论文集 , 2010, pp. 2613–2621.

[87] H. van Hasselt、 A. Guez 和 D. Silver, "深度强化学习与双 Q 学习," 在人工智能进步协会会议录，2016 年，第 2094-2100 页。

[88] A. Vezhnevets、 V. Mnih 、 S. Osindero 、 A. Graves 、 O. Vinyals 、 J. Agapiou 和 K. Kavukcuoglu。"用于学习宏观动作的战略性注意力写手," 在神经信息处理系统会议录，2016 年，第 3486-3494 页。

[89] A. S. Vezhnevets、 S. Osindero 、 T. Schaul 、 N. Heess 、 M. Jaderberg 、 D. Silver 和 K. Kavukcuoglu, "联邦网络用于层次强化学习," 在国际机器学习会议，2017 年，第 3540-3549 页。

[90] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "从图像像素中学习深度动态模型," in *IFAC* 系统辨识研讨会论文集 , 2015, pp. 1059–1064.

[91] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "从像素到力矩：使用深度动态模型的策略学习," in *ICML* 深度学习研讨会 , 2015.

[92] Z. Wang, N. de Freitas, and M. Lanctot, "深度强化学习的对抗网络架构," in 国际学习表征会议论文集 , 2016.

[93] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "具有经验重放的样本高效 Actor-Critic," in 国际学习表征会议论文集 , 2017.

[94] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[95] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Proc. Neural Information Processing Systems*, 2015, pp. 2746–2754.

[96] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber, "Recurrent policy gradients," *Logic J. IGPL*, vol. 18, no. 5, pp. 620–634, 2010.

[97] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learning*, vol. 8, no. 3–4, pp. 229–256, 1992.

[98] M. Wulfmeier, P. Ondruska, and I. Posner, "最大熵深度逆强化学习," 在 *NIPS* 深度强化学习研讨会 , 2015.

[99] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, 和 Y. Bengio, "展示、注意和讲述：具有视觉注意力的神经图像描述生成," 在 国际机器学习会议论文集 , 2015, pp. 2048–2057.

[100] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, 和 A. Farhadi, "使用深度强化学习的室内场景目标驱动视觉导航," 在 *IEEE* 国际机器人与自动化会议论文集 , 2017, pp. 3357–3364.