The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Rainbow: Combining Improvements in Deep Reinforcement Learning

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, David Silver
DeepMind

Abstract

The deep reinforcement learning community has made several independent improvements to the DQN algorithm. However, it is unclear which of these extensions are complementary and can be fruitfully combined. This paper examines six extensions to the DQN algorithm and empirically studies their combination. Our experiments show that the combination provides state-of-the-art performance on the Atari 2600 benchmark, both in terms of data efficiency and final performance. We also provide results from a detailed ablation study that shows the contribution of each component to overall performance.

Introduction

The many recent successes in scaling reinforcement learning (RL) to complex sequential decision-making problems were kick-started by the Deep Q-Networks algorithm (DQN; Mnih et al. 2013, 2015). Its combination of Q-learning with convolutional neural networks and experience replay enabled it to learn, from raw pixels, how to play many Atari games at human-level performance. Since then, many extensions have been proposed that enhance its speed or stability.

Double DQN (DDQN; van Hasselt, Guez, and Silver 2016) addresses an overestimation bias of Q-learning (van Hasselt 2010), by decoupling selection and evaluation of the bootstrap action. Prioritized experience replay (Schaul et al. 2015) improves data efficiency, by replaying more often transitions from which there is more to learn. The dueling network architecture (Wang et al. 2016) helps to generalize across actions by separately representing state values and action advantages. Learning from multi-step bootstrap targets (Sutton 1988; Sutton and Barto 1998), as used in A3C (Mnih et al. 2016), shifts the bias-variance tradeoff and helps to propagate newly observed rewards faster to earlier visited states. Distributional Q-learning (Bellemare, Dabney, and Munos 2017) learns a categorical distribution of discounted returns, instead of estimating the mean. Noisy DQN (Fortunato et al. 2017) uses stochastic network layers for exploration. This list is, of course, far from exhaustive.

Each of these algorithms enables substantial performance improvements in isolation. Since they address radically different issues, and since they build on a shared framework,

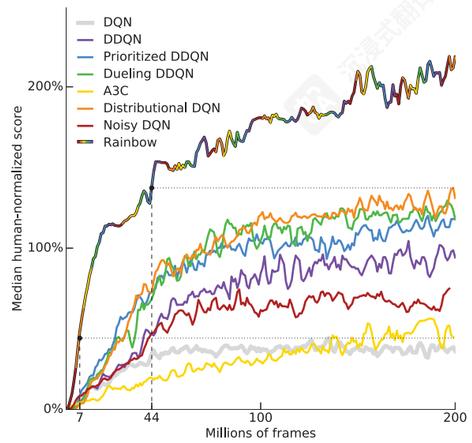

Figure 1: Median human-normalized performance across 57 Atari games. We compare Rainbow (rainbow-colored) to DQN and six published baselines. We match DQN's best performance after 7M frames, surpass any baseline in 44M frames, reaching substantially improved final performance. Curves are smoothed with a moving average of 5 points.

they could plausibly be combined. In some cases this has been done: Prioritized DDQN and Dueling DDQN both use double Q-learning, and Dueling DDQN was also combined with prioritized replay. In this paper we propose to study an agent that combines all the aforementioned ingredients. We show how these different ideas can be integrated, and that they are indeed complementary. In fact, their combination results in new state-of-the-art results on the benchmark suite of 57 Atari 2600 games from the Arcade Learning Environment (Bellemare et al. 2013), both in terms of data efficiency and of final performance. Finally, we show results from ablation studies to help understand the contributions of the individual components.

The Thirty-Second AAAI Conference
人工智能（AAAI-18）国际会议

Rainbow：结合深度强化学习中的改进

Matteo Hessel，Joseph Modayil，Hado van Hasselt，Tom Schaul，
Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, David Silver
DeepMind

摘要

深度强化学习社区对 DQN 算法进行了多项独立的改进。然而，这些扩展中哪些是互补的并且可以有效地结合尚不清楚。本文考察了 DQN 算法的六项扩展，并实证研究了它们的结合。我们的实验表明，这种结合在 Atari 2600 基准测试中提供了最先进的性能，无论是在数据效率还是最终性能方面。我们还提供了一份详细的消融研究结果，展示了每个组件对整体性能的贡献。

引言

近年来在将强化学习（RL）扩展到复杂的序列决策问题方面取得的许多成功，是由深度 Q 网络算法（DQN；Mnih 等人，2013，2015）启动的。它结合了 Q 学习、卷积神经网络和经验回放，使其能够从原始像素中学习，以人类水平的性能玩许多 Atari 游戏。自那时以来，已经提出了许多扩展，以提高其速度或稳定性。

双深度 Q 网络（DDQN；van Hasselt、Guez 和 Silver，2016）通过解耦引导动作的选择和评估来解决 Q 学习的过度估计偏差（van Hasselt，2010）。优先经验回放（Schaul 等人，2015）通过回放更多可以从中学到更多内容的转换来提高数据效率。双头网络架构（Wang 等人，2016）通过分别表示状态值和动作优势来帮助跨动作泛化。从多步引导目标中学习（Sutton，1988；Sutton 和 Barto，1998），如 A3C（Mnih 等人，2016）中使用的那样，会改变偏差 - 方差权衡，并有助于更快地将新观察到的奖励传播到早期访问的状态。分布式 Q 学习（Bellemare、Dabney 和 Munos，2017）学习折扣回报的类别分布，而不是估计均值。噪声 DQN（Fortunato 等人，2017）使用随机网络层进行探索。当然，这个列表远非详尽。

这些算法在单独使用时都能显著提升性能。由于它们针对的是截然不同的问题，并且它们都基于一个共享的框架，

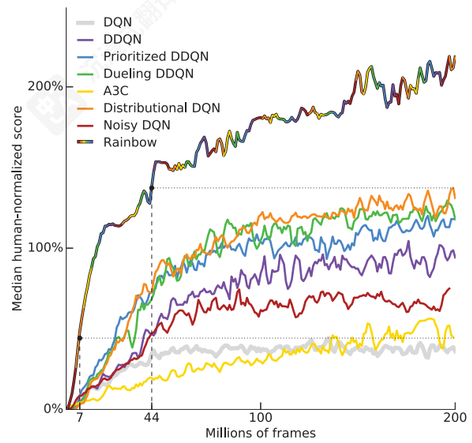

图 1：57 款 Atari 游戏中的人均归一化性能中位数。我们将 Rainbow（彩虹色）与 DQN 和六个已发表的基线进行比较。我们在 7M 帧后匹配了 DQN 的最佳性能，在 44M 帧后超越了任何基线，最终性能显著提升。曲线使用 5 点移动平均进行平滑。

它们有可能被结合起来。在某些情况下，这已经实现了：优先级 DDQN 和对抗 DDQN 都使用双 Q 学习，而对抗 DDQN 也与优先级回放结合。在本文中，我们提出研究一个结合所有上述成分的智能体。我们展示了这些不同想法如何被整合，以及它们确实互补。事实上，它们的结合在 Arcade Learning Environment（Bellemare 等人，2013 年）的 57 款 Atari 2600 游戏基准套件上取得了新的最先进结果，无论是在数据效率还是最终性能方面。最后，我们展示了消融实验的结果，以帮助理解各个组件的贡献。

Background

Reinforcement learning addresses the problem of an *agent* learning to act in an *environment* in order to maximize a scalar *reward* signal. No direct supervision is provided to the agent, for instance it is never directly told the best action.

Agents and environments. At each discrete time step $t = 0, 1, 2\ldots$, the environment provides the agent with an observation S_t, the agent responds by selecting an action A_t, and then the environment provides the next reward R_{t+1}, discount γ_{t+1}, and state S_{t+1}. This interaction is formalized as a *Markov Decision Process*, or MDP, which is a tuple $\langle \mathcal{S}, \mathcal{A}, T, r, \gamma \rangle$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $T(s, a, s') = P[S_{t+1} = s' \mid S_t = s, A_t = a]$ is the (stochastic) transition function, $r(s, a) = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$ is the reward function, and $\gamma \in [0, 1]$ is a discount factor. In our experiments MDPs will be *episodic* with a constant $\gamma_t = \gamma$, except on episode termination where $\gamma_t = 0$, but the algorithms are expressed in the general form.

On the agent side, action selection is given by a policy π that defines a probability distribution over actions for each state. From the state S_t encountered at time t, we define the discounted return $G_t = \sum_{k=0}^{\infty} \gamma_t^{(k)} R_{t+k+1}$ as the discounted sum of future rewards collected by the agent, where the discount for a reward k steps in the future is given by the product of discounts before that time, $\gamma_t^{(k)} = \prod_{i=1}^{k} \gamma_{t+i}$. An agent aims to maximize the expected discounted return by finding a good policy.

The policy may be learned directly, or it may be constructed as a function of some other learned quantities. In value-based reinforcement learning, the agent learns an estimate of the expected discounted return, or value, when following a policy π starting from a given state, $v^\pi(s) = E_\pi[G_t | S_t = s]$, or state-action pair, $q^\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a]$. A common way of deriving a new policy from a state-action value function is to act ϵ-greedily with respect to the action values. This corresponds to taking the action with the highest value (the *greedy* action) with probability $(1-\epsilon)$, and to otherwise act uniformly at random with probability ϵ. Policies of this kind are used to introduce a form of *exploration*: by randomly selecting actions that are sub-optimal according to its current estimates, the agent can discover and correct its estimates when appropriate. The main limitation is that it is difficult to discover alternative courses of action that extend far into the future; this has motivated research on more directed forms of exploration.

Deep reinforcement learning and DQN. Large state and/or action spaces make it intractable to learn Q value estimates for each state and action pair independently. In deep reinforcement learning, we represent the various components of agents, such as policies $\pi(s, a)$ or values $q(s, a)$, with deep (i.e., multi-layer) neural networks. The parameters of these networks are trained by gradient descent to minimize some suitable loss function.

In DQN (Mnih et al. 2015) deep networks and reinforcement learning were successfully combined by using a convolutional neural net to approximate the action values for a given state S_t (which is fed as input to the network in the form of a stack of raw pixel frames). At each step, based on the current state, the agent selects an action ϵ-greedily with respect to the action values, and adds a transition $(S_t, A_t, R_{t+1}, \gamma_{t+1}, S_{t+1})$ to a replay memory buffer (Lin 1992), that holds the last million transitions. The parameters of the neural network are optimized by using stochastic gradient descent to minimize the loss

$$(R_{t+1} + \gamma_{t+1} \max_{a'} q_{\overline{\theta}}(S_{t+1}, a') - q_\theta(S_t, A_t))^2 \,, \qquad (1)$$

where t is a time step randomly picked from the replay memory. The gradient of the loss is back-propagated only into the parameters θ of the *online network* (which is also used to select actions); the term $\overline{\theta}$ represents the parameters of a *target network*; a periodic copy of the online network which is not directly optimized. The optimization is performed using RMSprop (Tieleman and Hinton 2012), a variant of stochastic gradient descent, on mini-batches sampled uniformly from the experience replay. This means that in the loss above, the time index t will be a random time index from the last million transitions, rather than the current time. The use of experience replay and target networks enables relatively stable learning of Q values, and led to superhuman performance on several Atari games.

Extensions to DQN

DQN has been an important milestone, but several limitations of this algorithm are now known, and many extensions have been proposed. We propose a selection of six extensions that each have addressed a limitation and improved overall performance. To keep the size of the selection manageable, we picked a set of extensions that address distinct concerns (e.g., just one of the many addressing exploration).

Double Q-learning. Conventional Q-learning is affected by an overestimation bias, due to the maximization step in Equation 1, and this can harm learning. Double Q-learning (van Hasselt 2010), addresses this overestimation by decoupling, in the maximization performed for the bootstrap target, the selection of the action from its evaluation. It is possible to effectively combine this with DQN (van Hasselt, Guez, and Silver 2016), using the loss

$$(R_{t+1} + \gamma_{t+1} q_{\overline{\theta}}(S_{t+1}, \operatorname*{argmax}_{a'} q_\theta(S_{t+1}, a')) - q_\theta(S_t, A_t))^2 .$$

This change was shown to reduce harmful overestimations that were present for DQN, thereby improving performance.

Prioritized replay. DQN samples uniformly from the replay buffer. Ideally, we want to sample more frequently those transitions from which there is much to learn. As a proxy for learning potential, prioritized experience replay (Schaul et al. 2015) samples transitions with probability p_t relative to the last encountered absolute *TD error*:

$$p_t \propto \left| R_{t+1} + \gamma_{t+1} \max_{a'} q_{\overline{\theta}}(S_{t+1}, a') - q_\theta(S_t, A_t) \right|^\omega ,$$

where ω is a hyper-parameter that determines the shape of the distribution. New transitions are inserted into the replay

背景

强化学习解决了一个智能体如何在环境中行动以最大化一个标量奖励信号的问题。智能体不会得到直接的监督，例如，它永远不会被告知最佳行动。

智能体和环境。在每个离散的时间步 t =0，1,2…，环境为智能体提供一个观察 S_t，智能体通过选择一个行动 A_t 做出响应，然后环境提供下一个奖励 R_{t+1}、折扣 γ_{t+1} 和状态 S_{t+1}。这种交互被形式化为一个 马尔可夫决策过程，或 MDP，它是一个元组 $\langle \mathcal{S}, \mathcal{A}, T, r, \gamma \rangle$，其中 $\mathcal{S}$ 是一个有限的状态集，$\mathcal{A}$ 是一个有限的动作集，$T(s,a,s')$ $= P[S_{t+1} = s' \mid S_t = s, A_t = a]$是（随机的）转换函数，$r(s,a) = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$ 是奖励函数，而 $\gamma \in [0, 1]$ 是一个折扣因子。在我们的实验中，MDP 将是周期的，具有一个常数 $\gamma_t = \gamma$，除了在周期结束时 $\gamma_t = 0$，但算法以一般形式表示。

在智能体方面，动作选择由一个策略 π 给出，该策略定义了每个状态的动作概率分布。从时间 t 遇到的状态 S_t 开始，我们定义折扣回报 $G_t = \sum_{k=0}^{\infty} \gamma_t^{(k)} R_{t+k+1}$ 为智能体收集的未来奖励的折扣总和，其中未来 k 步的奖励折扣由该时间之前的折扣乘积 $\gamma_t^{(k)} = \prod_{i=1}^{k} \gamma_{t+i}$ 给出。智能体的目标是通过找到一个好的策略来最大化期望折扣回报。

策略可以直接学习，或者可以构建为其他学习量的函数。在基于值的强化学习中，智能体学习从给定状态 $v^\pi(s) = E_\pi[G_t | S_t = s]$ 或状态 - 动作对 $q^\pi(s,a) = E_\pi[G_t | S_t = s, A_t = a]$ 开始时策略 π 的期望折扣回报或值的估计。从状态 - 动作值函数导出新策略的一种常见方法是对动作值进行贪婪选择。这对应于以概率 $(1-\epsilon)$ 执行具有最高值（贪婪动作），并以概率 ϵ 以均匀随机的方式执行。这种策略用于引入一种形式的探索：通过随机选择根据其当前估计次优的动作，智能体可以在适当的时候发现和纠正其估计。主要限制是难以发现远期替代行动方案；这推动了更定向形式的探索研究。

深度强化学习和 DQN。大型状态和 / 或动作空间使得独立地为每个状态和动作对学习 Q 值估计变得难以处理。在深度强化学习中，我们使用深度（即多层）神经网络来表示智能体的各个组件，例如策略 $\pi(s,a)$ 或值 $q(s,a)$。这些网络的参数通过梯度下降来训练，以最小化某个合适的损失函数。

在 DQN （Mnih 等人，2015 年）中，深度网络和强化学习通过使用卷积神经网络来近似给定状态 {v1} （该状态以原始像素帧堆栈的形式输入到网络中）的动作值而被成功结合。

在每个步骤中，基于当前状态，智能体根据动作值贪婪地选择一个动作 ϵ，并将一个转换 $(S_t, A_t, R_{t+1}, \gamma_{t+1}, S_{t+1})$ 添加到回放内存缓冲区（ Lin 1992 ），该缓冲区保存了最后一百万个转换。通过使用随机梯度下降来优化神经网络的参数，以最小化损失

$$(R_{t+1} + \gamma_{t+1} \max_{a'} q_{\overline{\theta}}(S_{t+1}, a') - q_\theta(S_t, A_t))^2 \,, \qquad (1)$$

其中 t 是从回放内存中随机选择的一个时间步。损失的梯度仅反向传播到在线网络 θ 的参数中（该网络也用于选择动作）；术语 θ 表示目标网络的参数；目标网络是周期性地从在线网络复制过来的，不直接进行优化。优化使用 RMSprop （ Tieleman 和 Hinton 2012 年），随机梯度下降的一个变体，在从经验回放中均匀采样的小批量上进行。这意味着在上述损失中，时间索引 t 将是最后一百万次转换中的一个随机时间索引，而不是当前时间。经验回放和目标网络的使用使得 Q 值的学习相对稳定，并在多个 Atari 游戏中取得了超人类的表现。

Extensions to DQN

DQN has been an important milestone, but several limitations of this algorithm are now known, and many extensions have been proposed. We propose a selection of six extensions that each have addressed a limitation and improved overall performance. To keep the size of the selection manageable, we picked a set of extensions that address distinct concerns (e.g., just one of the many addressing exploration).

双 Q 学习。传统的 Q 学习受到公式 1 中最大化步骤的过度估计偏差影响，这会损害学习。双 Q 学习（ va n Hasselt 2010 ）通过在引导目标执行的最大化过程中解耦动作选择与其评估，解决了这种过度估计问题。可以有效地将其与 DQN （ van Hasselt, Guez, and Silver 2016 ）结合，使用损失

$$(R_{t+1} + \gamma_{t+1} q_{\overline{\theta}}(S_{t+1}, \operatorname*{argmax}_{a'} q_\theta(S_{t+1}, a')) - q_\theta(S_t, A_t))^2 .$$

这种变化被证明减少了 DQN 中存在的有害过度估计，从而提高了性能。

优先回放。 DQN 从回放缓冲区中均匀采样。理想情况下，我们希望更频繁地采样那些有很多可学习内容的转换。作为学习潜力的代理，优先经验回放（ Schaul et al. 2015 ）根据相对于上次遇到的绝对 *TD* 误差的概率 p_t 采样转换：

$$p_t \propto \left| R_{t+1} + \gamma_{t+1} \max_{a'} q_{\overline{\theta}}(S_{t+1}, a') - q_\theta(S_t, A_t) \right|^\omega ,$$

其中 ω 是一个决定分布形状的超参数。新的转换被插入到回放

buffer with maximum priority, providing a bias towards recent transitions. Note that stochastic transitions might also be favoured, even when there is little left to learn about them.

Dueling networks. The dueling network is a neural network architecture designed for value based RL. It features two streams of computation, the value and advantage streams, sharing a convolutional encoder, and merged by a special aggregator (Wang et al. 2016). This corresponds to the following factorization of action values:

$$q_\theta(s,a) = v_\eta(f_\xi(s)) + a_\psi(f_\xi(s),a) - \frac{\sum_{a'} a_\psi(f_\xi(s),a')}{N_{\text{actions}}},$$

where ξ, η, and ψ are, respectively, the parameters of the shared encoder f_ξ, of the value stream v_η, and of the advantage stream a_ψ; and $\theta = \{\xi, \eta, \psi\}$ is their concatenation.

Multi-step learning. Q-learning accumulates a single reward and then uses the greedy action at the next step to bootstrap. Alternatively, forward-view *multi-step* targets can be used (Sutton 1988). We define the truncated n-step return from a given state S_t as

$$R_t^{(n)} \equiv \sum_{k=0}^{n-1} \gamma_t^{(k)} R_{t+k+1}\,. \tag{2}$$

A multi-step variant of DQN is then defined by minimizing the alternative loss,

$$(R_t^{(n)} + \gamma_t^{(n)} \max_{a'} q_{\overline{\theta}}(S_{t+n},a') - q_\theta(S_t,A_t))^2.$$

Multi-step targets with suitably tuned n often lead to faster learning (Sutton and Barto 1998).

Distributional RL. We can learn to approximate the distribution of returns instead of the expected return. Recently Bellemare, Dabney, and Munos (2017) proposed to model such distributions with probability masses placed on a discrete support $\boldsymbol{z}$, where $\boldsymbol{z}$ is a vector with $N_{\text{atoms}} \in \mathbb{N}^+$ *atoms*, defined by $z^i = v_{\min} + (i-1)\frac{v_{\max}-v_{\min}}{N_{\text{atoms}}-1}$ for $i \in \{1,\dots,N_{\text{atoms}}\}$. The approximating distribution d_t at time t is defined on this support, with the probability mass $p_\theta^i(S_t,A_t)$ on each atom i, such that $d_t = (\boldsymbol{z}, \boldsymbol{p}_\theta(S_t,A_t))$. The goal is to update θ such that this distribution closely matches the actual distribution of returns.

To learn the probability masses, the key insight is that return distributions satisfy a variant of Bellman's equation. For a given state S_t and action A_t, the distribution of the returns under the optimal policy π^* should match a target distribution defined by taking the distribution for the next state S_{t+1} and action $a^*_{t+1} = \pi^*(S_{t+1})$, contracting it towards zero according to the discount, and shifting it by the reward (or distribution of rewards, in the stochastic case). A distributional variant of Q-learning is then derived by first constructing a new support for the target distribution, and then minimizing the Kullbeck-Leibler divergence between the distribution d_t and the target distribution $d'_t \equiv (R_{t+1} + \gamma_{t+1}\boldsymbol{z},\ \boldsymbol{p}_{\overline{\theta}}(S_{t+1}, \overline{a}^*_{t+1}))$,

$$D_{\text{KL}}(\Phi_{\boldsymbol{z}} d'_t || d_t)\,. \tag{3}$$

Here $\Phi_{\boldsymbol{z}}$ is a L2-projection of the target distribution onto the fixed support $\boldsymbol{z}$, and $\overline{a}^*_{t+1} = \text{argmax}_a\, q_{\overline{\theta}}(S_{t+1},a)$ is the greedy action with respect to the mean action values $q_{\overline{\theta}}(S_{t+1},a) = \boldsymbol{z}^\top \boldsymbol{p}_\theta(S_{t+1},a)$ in state S_{t+1}.

As in the non-distributional case, we can use a frozen copy of the parameters $\overline{\theta}$ to construct the target distribution. The parametrized distribution can be represented by a neural network, as in DQN, but with $N_{\text{atoms}} \times N_{\text{actions}}$ outputs. A *softmax* is applied independently for each action dimension of the output to ensure that the distribution for each action is appropriately normalized.

Noisy Nets. The limitations of exploring using ϵ-greedy policies are clear in games such as Montezuma's Revenge, where many actions must be executed to collect the first reward. Noisy Nets (Fortunato et al. 2017) propose a noisy linear layer that combines a deterministic and noisy stream,

$$\boldsymbol{y} = (\boldsymbol{b} + \mathbf{W}\boldsymbol{x}) + (\boldsymbol{b}_{noisy} \odot \epsilon^b + (\mathbf{W}_{noisy} \odot \epsilon^w)\boldsymbol{x}), \tag{4}$$

where ϵ^b and ϵ^w are random variables, and $\odot$ denotes the element-wise product. This transformation can then be used in place of the standard linear $\boldsymbol{y} = \boldsymbol{b} + \mathbf{W}\boldsymbol{x}$. Over time, the network can learn to ignore the noisy stream, but will do so at different rates in different parts of the state space, allowing state-conditional exploration with a form of self-annealing.

The Integrated Agent

In this paper we integrate all the aforementioned components into a single integrated agent, which we call *Rainbow*.

First, we replace the 1-step distributional loss (3) with a multi-step variant. We construct the target distribution by contracting the value distribution in S_{t+n} according to the cumulative discount, and shifting it by the truncated n-step discounted return. This corresponds to defining the target distribution as $d_t^{(n)} = (R_t^{(n)} + \gamma_t^{(n)}\boldsymbol{z},\ \ \boldsymbol{p}_{\overline{\theta}}(S_{t+n}, a^*_{t+n}))$. The resulting loss is

$$D_{\text{KL}}(\Phi_{\boldsymbol{z}} d_t^{(n)} || d_t)\,,$$

where, again, $\Phi_{\boldsymbol{z}}$ is the projection onto $\boldsymbol{z}$.

We combine the multi-step distributional loss with double Q-learning by using the greedy action in S_{t+n} selected according to the *online network* as the bootstrap action a^*_{t+n}, and evaluating such action using the *target network*.

In standard proportional prioritized replay (Schaul et al. 2015) the absolute TD error is used to prioritize the transitions. This can be computed in the distributional setting, using the mean action values. However, in our experiments all distributional Rainbow variants prioritize transitions by the KL loss, since this is what the algorithm is minimizing:

$$p_t \propto \left(D_{\text{KL}}(\Phi_{\boldsymbol{z}} d_t^{(n)} || d_t)\right)^\omega.$$

The KL loss as priority might be more robust to noisy stochastic environments because the loss can continue to decrease even when the returns are not deterministic.

The network architecture is a dueling network architecture adapted for use with return distributions. The network

具有最高优先级的缓冲区，为最近的转换提供偏好。请注意，随机转换也可能被偏爱，即使关于它们还有很少的东西可以学习了。

Dueling networks. Dueling network 是一种为基于值的 RL 设计的神经网络架构。它具有两个计算流，即值流和优势流，共享一个卷积编码器，并由一个特殊的聚合器合并（ Wang 等人，2016 年）。这对应于以下动作值的分解：

$$q_\theta(s,a) = v_\eta(f_\xi(s)) + a_\psi(f_\xi(s),a) - \frac{\sum_{a'} a_\psi(f_\xi(s),a')}{N_{\text{actions}}},$$

其中 ξ、 η 和 ψ 分别是共享编码器 f_ξ、值流 v_η 和优势流 a_ψ 的参数；而 $\theta = \{\xi, \eta, \psi\}$ 是它们的连接。

Multi-step learning. Q-learning 累积单个奖励，然后使用下一个步骤的贪婪动作来启动。或者，可以使用前瞻式多步目标（ Sutton，1988 年）。我们定义从给定状态 S_t 开始的截断 n- 步回报为

$$R_t^{(n)} \equiv \sum_{k=0}^{n-1} \gamma_t^{(k)} R_{t+k+1}\,. \tag{2}$$

通过最小化交替损失，定义了一种多步 DQN 变体，

$$(R_t^{(n)} + \gamma_t^{(n)} \max_{a'} q_{\overline{\theta}}(S_{t+n},a') - q_\theta(S_t,A_t))^2.$$

具有适当调谐的 n 的多步目标通常能带来更快的收敛速度（ Sutton and Barto 1998 ）。

分布式强化学习。 我们可以学习近似回报的分布，而不是期望回报。最近 Bellemare 、 Dabney 和 Munos （ 2017 ）提出用概率质量在离散支撑集 $\boldsymbol{z}$ 上建模这种分布，其中 $\boldsymbol{z}$ 是一个具有 $N_{\text{atoms}} \in \mathbb{N}^+$ 原子的向量，由 $z^i = v_{\min} + (i-1)\frac{v_{\max}-v_{\min}}{N_{\text{atoms}}-1}$ 定义，对于 $i \in \{1,\dots,N_{\text{atoms}}\}$。在时间 t 时，近似分布 d_t 定义在这个支撑集上，每个原子 i 上的概率质量 $p_\theta^i(S_t,A_t)$，使得 $d_t = (\boldsymbol{z}, \boldsymbol{p}_\theta(S_t,A_t))$。目标是更新 θ ，使得这个分布与实际回报分布紧密匹配。

为了学习概率质量，关键洞察在于回报分布满足 Bellman 方程的一个变体。对于给定的状态 S_t 和动作 A_t，在最优策略 π^* 下回报的分布应该与一个目标分布匹配，该目标分布是通过取下一个状态 S_{t+1} 和动作 $a^*_{t+1} = \pi^*(S_{t+1})$ 的分布，根据折扣将其收缩到零，并加上奖励（或随机情况下的奖励分布）。然后推导出分布式的 Q-learning，首先构建目标分布的新支撑集，然后最小化分布 d_t 和目标分布 $d'_t \equiv (R_{t+1} + \gamma_{t+1}\boldsymbol{z},\ \boldsymbol{p}_\theta(S_{t+1}, \underline{a}^*_{t+1}))$ 之间的

$$D_{\text{KL}}(\Phi_{\boldsymbol{z}} d'_t || d_t)\,. \tag{3}$$

这里 $\Phi_{\boldsymbol{z}}$ 是目标分布到固定支撑集 $\boldsymbol{z}$ 的 L2 投影，而 $a^*_{t+1} = \text{argmax}_a\, q_\theta(S_{t+1},a)$ 是状态 S_{t+1} 中相对于平均动作值 $q_\theta(S_{t+1},a) = \boldsymbol{z}^\top \boldsymbol{p}_\theta(S_{t+1},a)$ 的贪婪动作。

与分布无关的情况一样，我们可以使用参数 θ 的冻结副本来构建目标分布。参数化的分布可以像 DQN 一样由神经网络表示，但具有 $N_{\text{atoms}} \times N_{\text{actions}}$ 个输出。对输出的每个动作维度独立应用 *sof tmax* ，以确保每个动作的分布适当归一化。

噪声网络。 在蒙特祖马之复仇等游戏中，使用 ϵ- 贪婪策略进行探索的局限性很明显，其中必须执行许多动作才能获得第一个奖励。噪声网络（ Fortunato 等人，2017 年）提出了一种噪声线性层，该层结合了确定性和噪声流，

$$\boldsymbol{y} = (\boldsymbol{b} + \mathbf{W}\boldsymbol{x}) + (\boldsymbol{b}_{noisy} \odot \epsilon^b + (\mathbf{W}_{noisy} \odot \epsilon^w)\boldsymbol{x}), \tag{4}$$

其中 ϵ^b 和 ϵ^w 是随机变量，而 $\odot$ 表示逐元素乘积。然后可以使用此转换代替标准的线性 $\boldsymbol{y} = \boldsymbol{b} + \mathbf{W}\boldsymbol{x}$。随着时间的推移，网络可以学会忽略噪声流，但在状态空间的不同部分将以不同的速率这样做，从而允许状态条件探索和某种自退火。

集成智能体

在本文中，我们将上述所有组件集成到一个单一的集成智能体中，我们称之为 *Rainbow*。

首先，我们用多步变体替换了 1 步分布损失（ 3 ）。我们通过根据累积折扣收缩 S_{t+n} 中的值分布，并将其通过截断 n- 步折扣回报来偏移，来构建目标分布。这对应于将目标分布定义为 $d_t^{(n)} = (R_t^{(n)} + \gamma_t^{(n)}\boldsymbol{z},\ \boldsymbol{p}_\theta(S_{t+n}, a^*_{t+n}))$. 得到的损失是

$$D_{\text{KL}}(\Phi_{\boldsymbol{z}} d_t^{(n)} || d_t)\,,$$

其中，再次地 ,$\Phi_{\boldsymbol{z}}$ 是投影到 $\boldsymbol{z}$。

我们将多步分布损失与双 Q 学习相结合，通过使用在 S_{t+n} 中根据在线网络 选择的贪婪动作作为引导动作 a^*_{t+n}，并使用目标网络评估该动作。

在标准比例优先重放（ Schaul 等人，2015 年）中，绝对 TD 误差用于优先处理转换。这可以在分布设置中计算，使用平均动作值。然而，在我们的实验中，所有分布的 Rainbow 变体都通过 KL 损失优先处理转换，因为这就是算法最小化的内容：

$$p_t \propto \left(D_{\text{KL}}(\Phi_{\boldsymbol{z}} d_t^{(n)} || d_t)\right)^\omega.$$

KL 损失作为优先级可能对噪声随机环境更鲁棒，因为即使回报不是确定的，损失也能持续减少。

网络架构是一种针对回报分布进行适配的博弈网络架构。网络

has a shared representation $f_\xi(s)$, which is then fed into a value stream v_η with N_{atoms} outputs, and into an advantage stream a_ξ with $N_{\text{atoms}} \times N_{\text{actions}}$ outputs, where $a^i_\xi(f_\xi(s), a)$ will denote the output corresponding to atom i and action a. For each atom z^i, the value and advantage streams are aggregated, as in dueling DQN, and then passed through a softmax layer to obtain the normalised parametric distributions used to estimate the returns' distributions:

$$p^i_\theta(s,a) = \frac{\exp(v^i_\eta(\phi) + a^i_\psi(\phi,a) - \overline{a}^i_\psi(s))}{\sum_j \exp(v^j_\eta(\phi) + a^j_\psi(\phi,a) - \overline{a}^j_\psi(s))},$$

where $\phi = f_\xi(s)$ and $\overline{a}^i_\psi(s) = \frac{1}{N_{\text{actions}}} \sum_{a'} a^i_\psi(\phi, a')$.

We then replace all linear layers with their noisy equivalent described in Equation (4). Within these noisy linear layers we use factorised Gaussian noise (Fortunato et al. 2017) to reduce the number of independent noise variables.

Experimental Methods

We now describe the methods and setup used for configuring and evaluating the learning agents.

Evaluation Methodology. We evaluated all agents on 57 Atari 2600 games from the arcade learning environment (Bellemare et al. 2013). We follow the training and evaluation procedures of Mnih et al. (2015) and van Hasselt et al. (2016). The average scores of the agent are evaluated during training, every 1M steps in the environment, by suspending learning and evaluating the latest agent for 500K frames. Episodes are truncated at 108K frames (or 30 minutes of simulated play), as in van Hasselt et al. (2016).

Agents' scores are normalized, per game, so that 0% corresponds to a random agent and 100% to the average score of a human expert. Normalized scores can be aggregated across all Atari levels to compare the performance of different agents. It is common to track the *median* human normalized performance across all games. We also consider the number of games where the agent's performance is above some fraction of human performance, to disentangle where improvements in the median come from. The *mean* human normalized performance is potentially less informative, as it is dominated by a few games (e.g., Atlantis) where agents achieve scores orders of magnitude higher than humans do.

Besides tracking the median performance as a function of environment steps, at the end of training we re-evaluate the best agent snapshot using two different testing regimes. In the *no-ops starts* regime, we insert a random number (up to 30) of no-op actions at the beginning of each episode (as we do also in training). In the *human starts* regime, episodes are initialized with points randomly sampled from the initial portion of human expert trajectories (Nair et al. 2015); the difference between the two regimes indicates the extent to which the agent has over-fit to its own trajectories.

Hyper-parameter tuning. All Rainbow's components have a number of hyper-parameters. The combinatorial space of hyper-parameters is too large for an exhaustive search, therefore we have performed limited tuning. For each component, we started with the values used in the paper that introduced this component, and tuned the most sensitive among hyper-parameters by manual coordinate descent.

DQN and its variants do not perform learning updates during the first $200K$ frames, to ensure sufficiently uncorrelated updates. We have found that, with prioritized replay, it is possible to start learning sooner, after only $80K$ frames.

DQN starts with an exploration ϵ of 1, corresponding to acting uniformly at random; it anneals the amount of exploration over the first 4M frames, to a final value of 0.1 (lowered to 0.01 in later variants). Whenever using Noisy Nets, we acted fully greedily ($\epsilon = 0$), with a value of 0.5 for the σ_0 hyper-parameter used to initialize the weights in the noisy stream[1]. For agents without Noisy Nets, we used ϵ-greedy but decreased the exploration rate faster than was previously used, annealing ϵ to 0.01 in the first $250K$ frames.

We used the Adam optimizer (Kingma and Ba 2014), which we found less sensitive to the choice of the learning rate than RMSProp. DQN uses a learning rate of $\alpha = 0.00025$ In all Rainbow's variants we used a learning rate of $\alpha/4$, selected among $\{\alpha/2, \alpha/4, \alpha/6\}$, and a value of 1.5×10^{-4} for Adam's ϵ hyper-parameter.

The value of n in multi-step learning is a sensitive hyper-parameter of Rainbow. We compared values of $n = 1$, 3, and 5. We observed that both $n = 3$ and 5 did well initially, but overall $n = 3$ performed the best by the end. For replay prioritization we used the recommended proportional variant, with priority exponent ω of 0.5, and linearly increased the importance sampling exponent β from 0.4 to 1 over the course of training. The priority exponent ω was tuned comparing values of $\{0.4, 0.5, 0.7\}$. Using the KL loss of distributional DQN as priority, we have observed that performance is very robust to the choice of ω.

The hyper-parameters (see Table 1) are identical across all 57 games, i.e., the Rainbow agent really is a *single* agent setup that performs well across all the games.

[1]The noise was generated on the GPU. Tensorflow noise generation can be unreliable on GPU. If generating the noise on the CPU, lowering σ_0 to 0.1 may be helpful.

Parameter	Value
Min history to start learning	80K frames
Adam learning rate	0.0000625
Exploration ϵ	0.0
Noisy Nets σ_0	0.5
Target Network Period	32K frames
Adam ϵ	1.5×10^{-4}
Prioritization type	proportional
Prioritization exponent ω	0.5
Prioritization importance sampling β	$0.4 \rightarrow 1.0$
Multi-step returns n	3
Distributional atoms	51
Distributional min/max values	$[-10, 10]$

Table 1: Rainbow hyper-parameters

具有共享表示 $f_\xi(s)$，然后输入到一个具有 N_{atoms} 输出的值流 v_η 和一个具有 $N_{\text{atoms}} \times N_{\text{actions}}$ 输出的优势流 a_ξ 中，其中 $a^i_\xi(f_\xi(s)\ a)$ 将表示对应于原子 i 和动作 a 的输出。对于每个原子 z^i，值和优势流将被聚合，就像在 dueling DQN 中一样，然后通过一个 soft max 层以获得用于估计回报分布的标准化参数分布：

$$p^i_\theta(s,a) = \frac{\exp(v^i_\eta(\phi) + a^i_\psi(\phi,a) - \overline{a}^i_\psi(s))}{\sum_j \exp(v^j_\eta(\phi) + a^j_\psi(\phi,a) - \overline{a}^j_\psi(s))},$$

其中 $\phi = f_\xi(s)$ 和 $\overline{a}^i_\psi(s) = \frac{1}{N_{\text{actions}}} \sum_{a'} a^i_\psi(\phi, a')$.

然后我们将所有线性层替换为方程 (4) 中描述的它们的噪声等价物。在这些噪声线性层中，我们使用分解高斯噪声（Fortunato 等人，2017 年）来减少独立噪声变量的数量。

实验方法

我们现在描述用于配置和评估学习智能体的方法和设置。

评估方法。 我们在街机学习环境（Bellemare 等人，2013 年）中的 57 款 Atari 2600 游戏上评估了所有智能体。我们遵循 Mnih 等人（2015 年）和 van Hasselt 等人（2016 年）的训练和评估程序。智能体的平均分数在训练过程中进行评估，每 1M 步在环境中，通过暂停学习并评估最新的智能体 500K 帧。像 van Hasselt 等人（2016 年）一样，回合在 108K 帧处截断（或模拟 30 分钟的播放）。

智能体的分数按游戏进行归一化，使得 0% 对应随机智能体，100% 对应人类专家的平均分数。归一化分数可以跨所有 Atari 关卡进行汇总，以比较不同智能体的性能。通常跟踪所有游戏中的人类归一化中位数性能。我们还考虑智能体性能高于人类性能一定比例的游戏数量，以分离中位数改进的来源。人类归一化平均性能可能不太具有信息量，因为它被少数游戏（例如，亚特兰蒂斯）主导，在这些游戏中，智能体的分数比人类高几个数量级。

除了跟踪随着环境步数变化的中位数性能外，在训练结束时，我们使用两种不同的测试机制重新评估最佳代理快照。在 无操作启动 机制中，我们在每个回合的开始插入一个随机数（最多 30 个）的无操作。在 人类启动 机制中，回合使用从人类专家轨迹的初始部分随机采样的点初始化（Nair 等人，2015 年）；这两种机制之间的差异表明代理过度拟合其自身轨迹的程度。

超参数调整。 所有 Rainbow 的组件都有一些超参数。超参数的组合空间太大，无法进行穷举搜索，因此我们进行了有限的调整。对于每个组件，我们从原始论文中引入该组件时使用的值开始，并通过手动坐标下降调整了最敏感的超参数。

DQN 及其变体在第一 $200K$ 帧期间不执行学习更新，以确保足够不相关的更新。我们发现，使用优先重放，可以在仅 $80K$ 帧后更早地开始学习。

DQN 以 1 的探索率 ϵ 开始，对应于完全随机地行动；它在前 4M 帧内逐渐减少探索率，最终值为 0.1（在后续变体中降低为 0.01）。在每次使用 Noisy Nets 时，我们完全贪婪地行动（$\epsilon = 0$），并使用值为 0.5 的 σ_0 超参数来初始化噪声流[1]中的权重。对于没有 Noisy Nets 的智能体，我们使用了 ϵ- 贪婪策略，但比之前更快地减少了探索率，在前 $250K$ 帧内将探索率退火到 0.01。

我们使用了 Adam 优化器（Kingma 和 Ba 2014），发现它对学习率的选择不如 RMSProp 敏感。DQN 使用学习率为 α =0.00025 。在所有 Rainbow 的变体中，我们使用了学习率为 $\alpha/4$，从 $\{\alpha/2, \alpha/4, \alpha/6\}$ 中选择，并使用值为 1.5×10^{-4} 的 Adam 的 ϵ 超参数。

n 在多步学习中的值是 Rainbow 的一个敏感超参数。我们比较了 n =1 、 3 和 5 的值。我们观察到 $n = 3$ 和5 在初期表现都很好，但最终 $n = 3$ 表现最佳。对于重放优先级，我们使用了推荐的比例变体，优先指数 ω 为 0.5，并将重要性采样指数 β 从 0.4 线性增加到 1。优先指数 ω 通过比较 $\{0.4, 0.5, 0.7\}$ 的值进行调优。使用分布式 DQN 的 KL 损失作为优先级，我们观察到性能对 ω 的选择非常鲁棒。

超参数（见表 1）在所有 57 局游戏中都是相同的，即 Rainbow 代理确实是一个在所有游戏中表现良好的单一 代理设置。

[1]噪声在 GPU 上生成。TensorFlow 在 GPU 上的噪声生成可能不可靠。如果在 CPU 上生成噪声，将 σ_0 降低到 0.1 可能有助于改善。

参数	Value
开始学习的最小历史记录	80K 帧 s
Adam 学习率	0.0000625
探索 ϵ	0.0
噪声网络 σ_0	0.5
目标网络周期	32K 帧 s
Adam ϵ	1.5×10^{-4}
Prioritization type	proportional
Prioritization exponent ω	0.5
Prioritization importance sampling β	$0.4 \rightarrow 1.0$
Multi-step returns n	3
Distributional atoms	51
Distributional min/max values	$[-10, 10]$

表 1：Rainbow 超参数

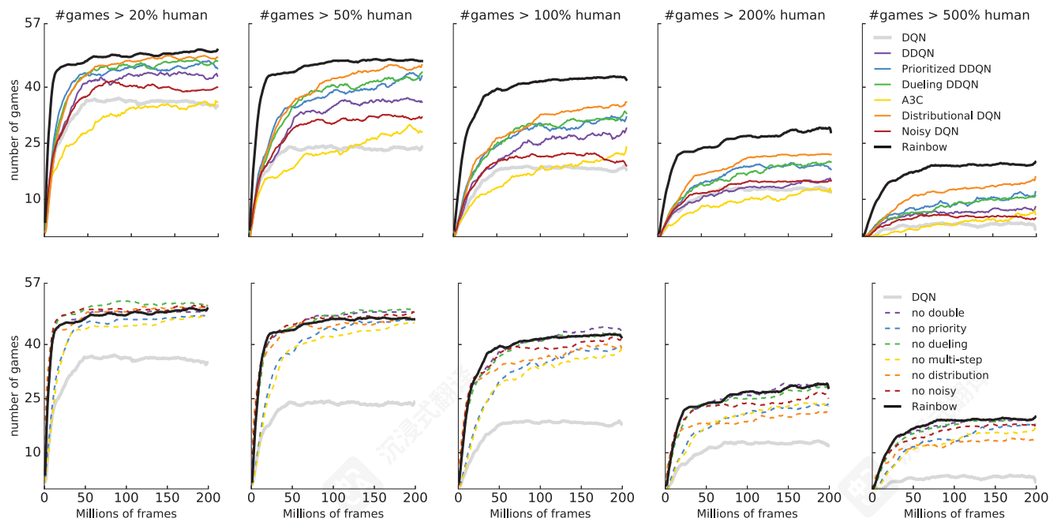

Figure 2: Each plot shows, for several agents, the number of games where they have achieved at least a given fraction of human performance, as a function of time. From left to right we consider the 20%, 50%, 100%, 200% and 500% thresholds. On the first row we compare Rainbow to the baselines. On the second row we compare Rainbow to its ablations.

Analysis

In this section we analyse the main experimental results. First, we show that Rainbow compares favorably to several published agents. Then we perform ablation studies, comparing several variants of the agent, each corresponding to removing a single component from Rainbow.

Comparison to published baselines. In Figure 1 we compare the Rainbow's performance (measured in terms of the median human normalized score across games) to the corresponding curves for A3C, DQN, DDQN, Prioritized DDQN, Dueling DDQN, Distributional DQN, and Noisy DQN. We thank the authors of the Dueling and Prioritized agents for providing the learning curves of these, and report our own re-runs for DQN, A3C, DDQN, Distributional DQN and Noisy DQN. The performance of Rainbow is better than any of the baselines by a large margin, both in data efficiency, as well as in final performance. Note that we match final performance of DQN after 7M frames, surpass the best final performance of these baselines in 44M frames, and reach substantially improved final performance.

In the final evaluations of the agent, after the end of training, Rainbow achieves a median score of 231% in the no-ops regime; in the human starts regime we measured a median score of 153%. In Table 2 we compare these scores to the published median scores of the individual baselines.

In Figure 2 (top row) we plot the number of games where an agent has reached some specified level of human normalized performance. From left to right, the subplots show on how many games the different agents have achieved at least 20%, 50%, 100%, 200% and 500% human normalized performance. This allows us to identify where the overall improvements in performance come from. Note that the gap in performance between Rainbow and other agents is apparent at all levels of performance: the Rainbow agent is improving scores on games where the baseline agents were already good, as well as improving in games where baseline agents are still far from human performance.

Agent	no-ops	human starts
DQN	79%	68%
DDQN (*)	117%	110%
Prioritized DDQN (*)	140%	128%
Dueling DDQN (*)	151%	117%
A3C (*)	-	116%
Noisy DQN	118%	102%
Distributional DQN	185%	125%
Rainbow	231%	153%

Table 2: Median normalized scores of the best agent snapshots for Rainbow and baselines. For methods marked with an asterisk, the scores come from the corresponding publication. DQN's scores comes from the dueling networks paper, since DQN's paper did not report scores for all 57 games. The others scores come from our own implementations.

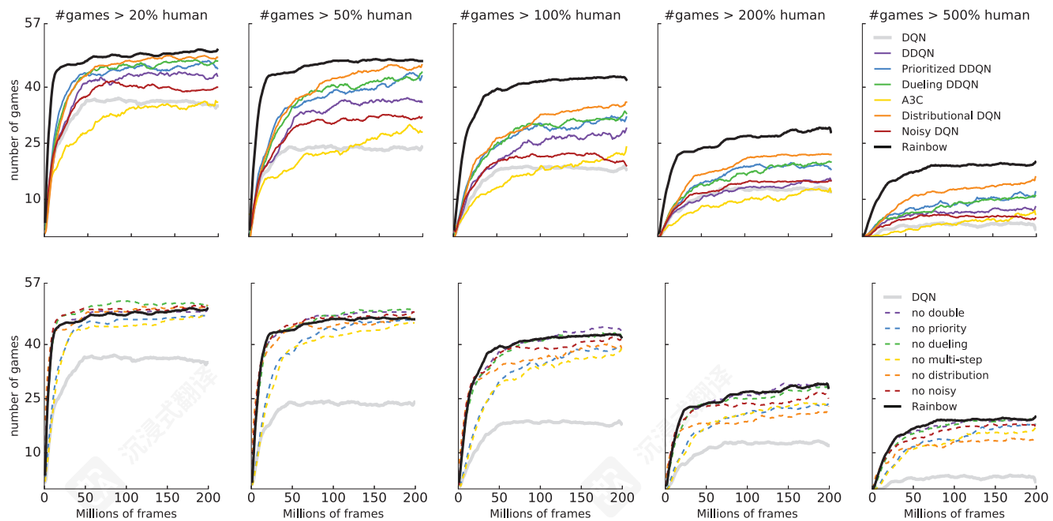

图 2：每个图展示了多个智能体在达到至少给定比例的人类表现水平的游戏数量随时间的变化。从左到右我们考虑了 20%、50%、100%、200% 和 500% 的阈值。在第一行我们比较了 Rainbow 与基线模型。在第二行我们比较了 Rainbow 与其消融实验结果。

分析

在本节中我们分析了主要的实验结果。首先，我们展示 Rainbow 与多个已发表的智能体相比表现优异。然后我们进行消融实验，比较了智能体的多个变体，每个变体对应从 Rainbow 中移除一个组件。

与已发表的基线模型比较。 在图 1 中我们比较了 Rainbow 的性能（以跨游戏的 median human normalized score 衡量）与 A3C、DQN、DDQN、Prioritized DDQN、Dueling DDQN、Distributional DQN 和 Noisy DQN 的对应曲线。我们感谢 Dueling 和 Prioritized 智能体的作者提供了这些的学习曲线，并报告了我们自己的 DQN、A3C、DDQN、Distributional DQN 和 Noisy DQN 的重新运行结果。Rainbow 的性能在数据效率和最终性能方面都显著优于所有基线模型。请注意，我们在 7M 帧 DQN 达到最终性能后匹配了最终性能，在 44M 帧时超越了这些基线模型的最佳最终性能，并达到了显著提升的最终性能。

在智能体的最终评估中，训练结束后，Rainbow 在 no-ops 状态下达到了 231% 的中位数分数；在 human starts 状态下，我们测量了 153% 的中位数分数。在表 2 中，我们将这些分数与各个基线的已发布中位数分数进行了比较。

In Figure 2 (top row) we plot the number of games where an agent has reached some specified level of human normalized performance. From left to right, the subplots show on 不同智能体在至少达到 20%、50%、100%、200% 和 500% 人类标准化表现的游戏数量。这使我们能够确定整体性能提升的来源。请注意，Rainbow 与其他智能体之间的性能差距在所有性能水平上都很明显：Rainbow 智能体不仅在与基线智能体表现已经很好的游戏中提升分数，还在基线智能体表现远未达到人类水平的游戏中进行提升。

智能体	no-ops	human starts
DQN	79%	68%
DDQN (*)	117%	110%
优先级 DDQN (*)	140%	128%
博弈 DDQN (*)	151%	117%
A3C (*)	-	116%
噪声 DQN	118%	102%
分布式 DQN	185%	125%
Rainbow	231%	153%

表 2：Rainbow 和基线方法的最佳代理快照的中值归一化分数。对于用星号标记的方法，这些分数来自相应的公开出版物。DQN 的分数来自斗篷网络论文，因为 DQN 的论文没有报告所有 57 个游戏的分数。其他分数来自我们自己的实现。

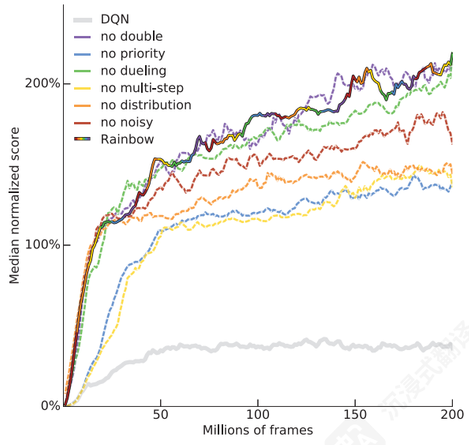

Figure 3: Median human-normalized performance across 57 Atari games, as a function of time. We compare our integrated agent (rainbow-colored) to DQN (gray) and to six different ablations (dashed lines). Curves are smoothed with a moving average over 10 points.

Learning speed. As in the original DQN setup, we ran each agent on a single GPU. The 7M frames required to match DQN's final performance correspond to less than 10 hours of wall-clock time. A full run of 200M frames corresponds to approximately 10 days, and this varies by less than 20% between all of the discussed variants. The literature contains many alternative training setups that improve performance as a function of wall-clock time by exploiting parallelism, e.g., Nair et al. (2015), Salimans et al. (2017), and Mnih et al. (2016). Properly relating the performance across such very different hardware/compute resources is non-trivial, so we focused exclusively on algorithmic variations, allowing apples-to-apples comparisons. While we consider them to be important and complementary, we leave questions of scalability and parallelism to future work.

Ablation studies. Since Rainbow integrates several different ideas into a single agent, we conducted additional experiments to understand the contribution of the various components, in the context of this specific combination.

First, we performed an ablation study. In each ablation, we removed a single component from the full Rainbow combination, and trained the resulting agent on all Atari games. Figure 3 compares median normalized scores of Rainbow to the six ablated variants. Figure 2 (bottom row) shows a more detailed breakdown of how these ablations perform relative to different thresholds of human normalized performance, and Figure 4 shows the gain or loss from each ablation for every game, averaged over the full learning run.

Prioritized replay and multi-step learning were the two most crucial components of Rainbow, in that removing either component caused a large drop in median performance. Unsurprisingly, the removal of either of these hurt early performance. Perhaps more surprisingly, the removal of multi-step learning also hurt final performance. Zooming in on individual games (Figure 4), we see both components helped almost uniformly across games (Rainbow performed better than either ablation in 53 games out of 57).

Distributional Q-learning ranked immediately below the previous techniques for relevance to the agent's performance. Notably, in early learning no difference is apparent, as shown in Figure 3, where for the first 40 million frames the distributional-ablation performed as well as the full agent. However, without distributions, the performance of the agent then started lagging behind. When the results are separated relatively to human performance in Figure 2, we see that the distributional-ablation primarily seems to lags on games that are above human level or near it.

In terms of median performance, the agent performed better when Noisy Nets were included; when these are removed and exploration is delegated to the traditional ϵ-greedy mechanism, performance was worse in aggregate (red line in Figure 3). While the removal of Noisy Nets produced a large drop in performance for several games, it also provided small increases in other games (Figure 4).

In aggregate, we did not observe a significant difference when removing the dueling network from the full Rainbow. The median score, however, hides the fact that the impact of Dueling differed between games, as shown by Figure 4. Figure 2 shows that Dueling perhaps provided some improvement on games with above-human performance levels (# games $>$ 200%), and some degradation on games with sub-human performance (# games $>$ 20%).

Also in the case of double Q-learning, the observed difference in median performance (Figure 3) is limited, with the component sometimes harming or helping depending on the game (Figure 4). To further investigate the role of double Q-learning, we compared the predictions of our trained agents to the actual discounted returns computed from clipped rewards. Comparing Rainbow to the agent where double Q-learning was ablated, we observed that the actual returns are often higher than 10 and therefore fall outside the support of the distribution, spanning from -10 to $+10$. This leads to underestimated returns, rather than overestimations. We hypothesize that clipping the values to this constrained range counteracts the overestimation bias of Q-learning. Note, however, that the importance of double Q-learning may increase if the support of the distributions is expanded.

Discussion

We have demonstrated that several improvements to DQN can be successfully integrated into a single learning algorithm that achieves state-of-the-art performance. Moreover, we have shown that within the integrated algorithm, all but one of the components provided clear performance benefits. There are many more algorithmic components that we were not able to include, which would be promising candi-

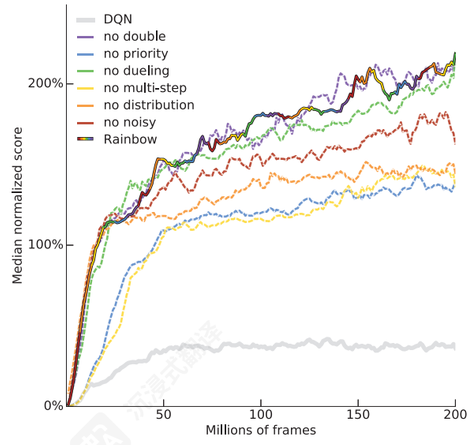

Figure 3: Median human-normalized performance across 57 Atari games, as a function of time. We compare our integrated agent (rainbow-colored) to DQN (gray) and to six different ablations (dashed lines). Curves are smoothed with a moving average over 10 points.

学习速度。与原始 DQN 设置一样，我们每个代理都在单个 GPU 上运行。要达到 DQN 最终性能所需的 7 M 帧对应于不到 10 小时的墙钟时间。200M 帧的完整运行对应于大约 10 天，并且所有讨论的变体之间变化小于 20%。文献中包含许多利用并行性来提高作为墙钟时间函数的性能的替代训练设置，例如 Nair 等人（ 2015 年）、 Salimans 等人（ 2017 年）和 Mnih 等人（ 2016 年）。正确地关联这种非常不同的硬件 / 计算资源之间的性能是非平凡的，因此我们专注于算法变化，允许苹果对苹果的比较。虽然我们认为它们很重要且互补，但我们把可扩展性和并行性的问题留给未来的工作。

消融研究。由于 Rainbow 将多种不同的思想集成到一个智能体中，我们进行了额外的实验来理解各种组件在这次特定组合中的贡献。

首先，我们进行了消融研究。在每个消融实验中，我们从完整的 Rainbow 组合中移除一个组件，并在所有 Atari 游戏上训练得到的智能体。图 3 比较了 Rainbow 与六个消融变体的中位数归一化分数。图 2 （底部行）展示了这些消融实验相对于不同人类归一化性能阈值的更详细表现，图 4 展示了每个游戏从每个消融实验中获得的增益或损失，这些数据是在整个学习运行中平均得到的。

优先重放和多步学习是 Rainbow 最重要的两个组件，因为移除任何一个组件都会导致中位数性能大幅下降。毫不奇怪，移除这两个组件中的任何一个都会损害早期性能。或许更令人惊讶的是，移除多步学习也会损害最终性能。通过查看单个游戏（图 4 ），我们看到这两个组件在大多数游戏中都起到了帮助作用（ Rainbow 在 57 场游戏中的 53 场游戏里表现优于消融实验）。

分布式 Q 学习在相关性方面立即排在先前技术之下。值得注意的是，在早期学习中没有明显差异，如图 3 所示，在前 4 亿帧内，分布式移除版本的表现与完整代理相当。然而，没有分布式时，代理的性能开始落后。当结果相对于人类性能在图 2 中分离时，我们看到分布式移除版本主要在高于人类水平或接近人类水平的游戏中落后。

在中位数性能方面，当包含噪声网络时，代理表现更好；当移除这些网络并将探索委托给传统的 ϵ- 贪婪机制时，整体性能更差（图 3 中的红线）。虽然移除噪声网络导致几个游戏性能大幅下降，但它也使其他一些游戏性能略有提升（图 4 ）。

总体而言，当我们从完整的 Rainbow 中移除决斗网络时，我们没有观察到显著差异。然而，中位数分数掩盖了这样一个事实：决斗网络对游戏的影响因游戏而异，如图 4 所示。图 2 显示，决斗网络可能对具有超人类表现水平的游戏（ # 游戏占 $>$ 200% ）提供了一些改进，而对表现低于人类的游戏（ # 游戏占 $>$ 20% ）造成了一些退化。

在双 Q 学习的情况下，观察到的中位数性能差异（图 3 ）也是有限的，该组件有时会根据游戏的不同而损害或帮助（图 4 ）。为了进一步研究双 Q 学习的作用，我们将我们训练代理的预测与从裁剪奖励计算的实际折扣回报进行了比较。将 Rainbow 与双 Q 学习被移除的代理进行比较时，我们观察到实际回报经常高于 10，因此超出了分布的支持范围，跨越从 -10 到 $+10$。这导致了回报的低估，而不是高估。我们假设将值裁剪到这个受限范围内可以抵消 Q 学习的过高估计偏差。然而，需要注意的是，如果分布的支持范围扩大，双 Q 学习的重要性可能会增加。

讨论

我们已经证明，对 DQN 的几种改进可以成功地集成到一个实现最先进性能的学习算法中。此外，我们已经表明，在集成算法中，除了一个组件外，所有组件都提供了明确性能优势。我们未能包括的还有许多其他算法组件，这些组件将是有希望的候选者

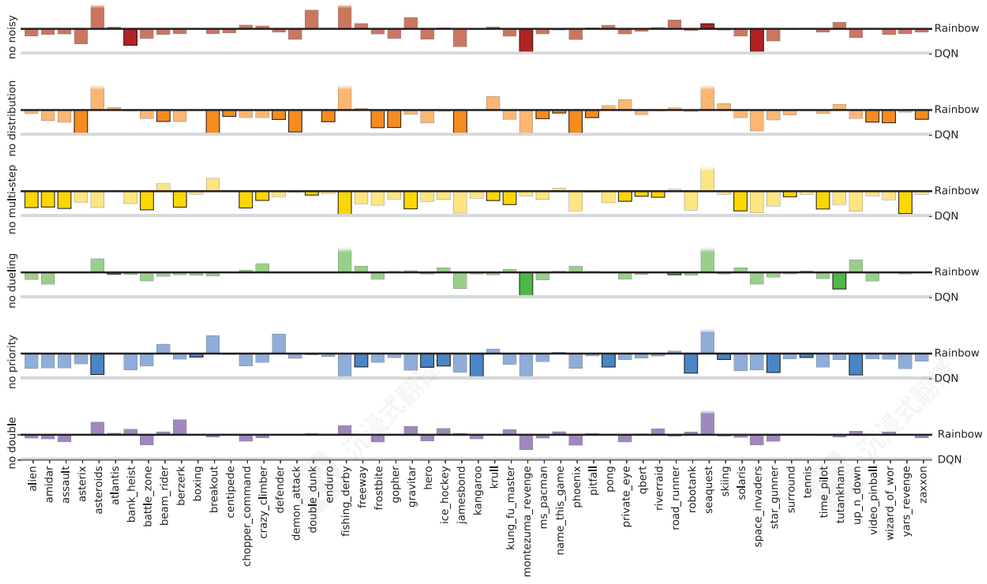

Figure 4: Performance drops of ablation agents: Performance is area under the curve, normalized relative to Rainbow and DQN. Venture and Bowling, where DQN outperformed Rainbow, are omitted. Ablations leading to the largest drop are highlighted per game. Prioritization and multi-step help almost uniformly across games, other components' impact differs per game.

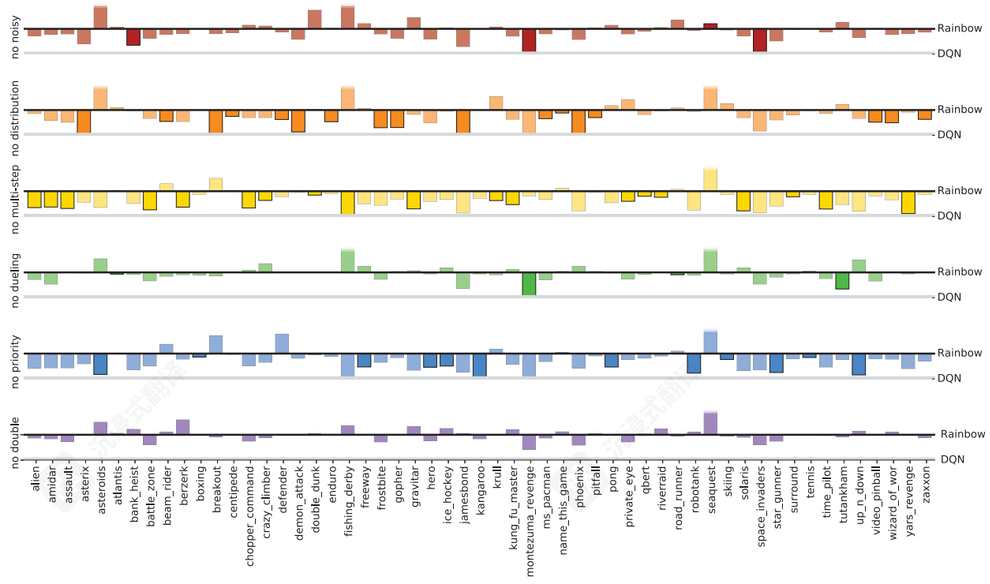

图 4：消融代理的性能下降：性能是曲线下面积，相对于 Rainbow 和 DQN 进行归一化。Venture 和 Bowling （其中 DQN 表现优于 Rainbow ）被省略。每个游戏中导致最大下降的消融操作被突出显示。优先级和多步帮助几乎在所有游戏中都有效，其他组件的影响因游戏而异。

dates for further experiments on integrated agents. Among the many possible candidates, we discuss several below.

We focused on value-based methods in the Q-learning family, but similar ideas may benefit also policy-based RL algorithms such as TRPO (Schulman et al. 2015), or actor-critic methods (Mnih et al. 2016; O'Donoghue et al. 2016).

A number of algorithms exploit sequences of data to improve learning efficiency. Optimality tightening (He et al. 2016) uses multi-step returns to construct additional inequality bounds, instead of just replacing the 1-step targets in Q-learning. Eligibility traces allow a soft combination over n-step returns (Sutton 1988). However, sequential methods all leverage more computation per update than the multi-step targets used in Rainbow. Also, the combination of prioritized replay with sequence data is still an open problem.

Episodic control (Blundell et al. 2016) also focuses on data efficiency, and was shown to be very effective in some domains. It improves early learning by using episodic memory as a complementary learning system, capable of immediately re-enacting successful action sequences.

Besides Noisy Nets, many exploration methods have been proposed: e.g. Bootstrapped DQN (Osband et al. 2016), intrinsic motivation (Stadie, Levine, and Abbeel 2015) and count-based exploration (Bellemare et al. 2016). Combining these with Rainbow is fruitful subject for further research.

We focused on the core learning updates, without exploring alternative computational architectures. Asynchronous learning from parallel copies of the environment, as in A3C (Mnih et al. 2016), Gorila (Nair et al. 2015), or Evolution Strategies (Salimans et al. 2017), can speed up learning in wall-clock time, although at the cost of data efficiency.

Hierarchical RL has also been applied with success to several complex Atari games. Among successful applications of HRL we highlight h-DQN (Kulkarni et al. 2016a) and Feudal Networks (Vezhnevets et al. 2017).

The state representation could be improved through the use of auxiliary tasks such as pixel or feature control (Jaderberg et al. 2016), supervised predictions (Dosovitskiy and Koltun 2016) or successor features (Kulkarni et al. 2016b).

To evaluate Rainbow fairly against baselines, we followed the common domain modifications of frame-stacking, reward clipping, and fixed action-repetition. These may be replaced by more principled techniques. Recurrent networks (Hausknecht and Stone 2015) can learn temporal state representations, replacing frame-stacking. Pop-Art (van Hasselt et al. 2016) enables learning from raw rewards. Fine-grained action repetition (Sharma, Lakshminarayanan, and Ravindran 2017) learns the number of action repetitions. In general, we believe that exposing the real game to agents is a promising direction for future research.

References

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation plat-

进一步实验集成代理的日期。在许多可能的候选者中，我们讨论了以下几个。

我们专注于 Q 学习家族中的基于值的方法，但类似的想法也可能受益于基于策略的 RL 算法，如 TRPO （ Schulman 等人，2015 年），或 Actor-Critic 方法（ Mnih 等人，2016 年；O' Donoghue 等人，2016 年）。

一些算法利用数据序列来提高学习效率。最优性收紧（ He 等人，2016 年）使用多步回报来构建额外的不等式界限，而不是仅仅替换 Q 学习中的 1 步目标。资格迹允许对 n 步回报进行软组合（ Sutton，1988 年）。然而，序列方法在每个更新中都利用比 Rainbow 中使用的多步目标更多的计算。此外，优先级重放与序列数据的结合仍然是一个开放的问题。

情景控制（ Blundell 等人 2016 年）也关注数据效率，并且在某些领域被证明非常有效。它通过使用情景记忆作为辅助学习系统来改进早期学习，该系统能够立即重演成功的动作序列。

除了噪声网络之外，还提出了许多探索方法：例如引导式 DQN （ Osband 等人 2016 年）、内在动机（ Stadie 、Levine 和 Abbeel 2015 年）和基于计数的探索（ Bellemare 等人 2016 年）。将这些与 Rainbow 结合起来是进一步研究的有益课题。

我们专注于核心学习更新，而没有探索替代的计算架构。异步从环境的并行副本中学习，如在 A3C （ Mnih 等人 2016 年）、Gorila （ Nair 等人 2015 年）或进化策略（ Salimans 等人 2017 年）中，可以在墙上时间加速学习，尽管以牺牲数据效率为代价。

分层强化学习（ Hierarchical RL ）也已成功应用于几种复杂的 Atari 游戏。在分层强化学习的成功应用中，我们重点介绍了 h-DQN （ Kulkarni 等人，2016a ）和联邦网络（ Feudal Networks ）（ Vezhnevets 等人，2017 ）。

状态表示可以通过使用辅助任务来改进，例如像素或特征控制（ Jaderberg 等人，2016 ）、监督预测（ Dosovitskiy 和 Koltun，2016 ）或后继特征（ Kulkarni 等人，2016b ）。

为了公平地评估 Rainbow 相对于基线的性能，我们遵循了常见的领域修改，包括帧堆叠、奖励裁剪和固定动作重复。这些可以通过更原则性的技术来替代。循环网络（ Hausknecht 和 Stone，2015 ）可以学习时间状态表示，替代帧堆叠。Pop-Art （ van Hasselt 等人，2016 ）能够从原始奖励中学习。细粒度动作重复（ Sharma 、Lakshminarayanan 和 Ravindran，2017 ）学习动作重复的次数。总的来说，我们认为将真实游戏暴露给智能体是未来研究的一个有前景的方向。

参考文献

Bellemare, M. G.; Naddaf, Y.; Veness, J.; 和 Bowling, M. 2013. 街机学习环境：一个评估平台

form for general agents. *J. Artif. Intell. Res. (JAIR)* 47:253–279.

Bellemare, M. G.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; and Munos, R. 2016. Unifying count-based exploration and intrinsic motivation. In *NIPS*.

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A distributional perspective on reinforcement learning. In *ICML*.

Blundell, C.; Uria, B.; Pritzel, A.; Li, Y.; Ruderman, A.; Leibo, J. Z.; Rae, J.; Wierstra, D.; and Hassabis, D. 2016. Model-Free Episodic Control. *ArXiv e-prints*.

Dosovitskiy, A., and Koltun, V. 2016. Learning to act by predicting the future. *CoRR* abs/1611.01779.

Fortunato, M.; Azar, M. G.; Piot, B.; Menick, J.; Osband, I.; Graves, A.; Mnih, V.; Munos, R.; Hassabis, D.; Pietquin, O.; Blundell, C.; and Legg, S. 2017. Noisy networks for exploration. *CoRR* abs/1706.10295.

Hausknecht, M., and Stone, P. 2015. Deep recurrent Q-learning for partially observable MDPs. *arXiv preprint arXiv:1507.06527*.

He, F. S.; Liu, Y.; Schwing, A. G.; and Peng, J. 2016. Learning to play in a day: Faster deep reinforcement learning by optimality tightening. *CoRR* abs/1611.01606.

Jaderberg, M.; Mnih, V.; Czarnecki, W. M.; Schaul, T.; Leibo, J. Z.; Silver, D.; and Kavukcuoglu, K. 2016. Reinforcement learning with unsupervised auxiliary tasks. *CoRR* abs/1611.05397.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Kulkarni, T. D.; Narasimhan, K.; Saeedi, A.; and Tenenbaum, J. B. 2016a. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *CoRR* abs/1604.06057.

Kulkarni, T. D.; Saeedi, A.; Gautam, S.; and Gershman, S. J. 2016b. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*.

Lin, L.-J. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning* 8(3):293–321.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing atari with deep reinforcement learning. *CoRR* abs/1312.5602.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*.

Nair, A.; Srinivasan, P.; Blackwell, S.; Alcicek, C.; Fearon, R.; De Maria, A.; Panneershelvam, V.; Suleyman, M.; Beattie, C.; Petersen, S.; Legg, S.; Mnih, V.; Kavukcuoglu, K.; and Silver, D. 2015. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*.

O'Donoghue, B.; Munos, R.; Kavukcuoglu, K.; and Mnih, V. 2016. Pgq: Combining policy gradient and q-learning. *CoRR* abs/1611.01626.

Osband, I.; Blundell, C.; Pritzel, A.; and Roy, B. V. 2016. Deep exploration via bootstrapped dqn. In *NIPS*.

Salimans, T.; Ho, J.; Chen, X.; and Sutskever, I. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR* abs/1703.03864.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized experience replay. In *Proc. of ICLR*.

Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.; and Abbeel, P. 2015. Trust region policy optimization. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, 1889–1897. JMLR.org.

Sharma, S.; Lakshminarayanan, A. S.; and Ravindran, B. 2017. Learning to repeat: Fine grained action repetition for deep reinforcement learning. *arXiv preprint arXiv:1702.06054*.

Stadie, B. C.; Levine, S.; and Abbeel, P. 2015. Incentivizing exploration in reinforcement learning with deep predictive models. *CoRR* abs/1507.00814.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. The MIT press, Cambridge MA.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine learning* 3(1):9–44.

Tieleman, T., and Hinton, G. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2):26–31.

van Hasselt, H.; Guez, A.; Guez, A.; Hessel, M.; Mnih, V.; and Silver, D. 2016. Learning values across many orders of magnitude. In *Advances in Neural Information Processing Systems 29*, 4287–4295.

van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double Q-learning. In *Proc. of AAAI*, 2094–2100.

van Hasselt, H. 2010. Double Q-learning. In *Advances in Neural Information Processing Systems 23*, 2613–2621.

Vezhnevets, A. S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; and Kavukcuoglu, K. 2017. Feudal networks for hierarchical reinforcement learning. *CoRR* abs/1703.01161.

Wang, Z.; Schaul, T.; Hessel, M.; van Hasselt, H.; Lanctot, M.; and de Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, 1995–2003.

通用智能体的表单。 *J. Artif . Intell. Res. (JAIR)*47:253– 279。

Bellemare, M. G.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; and Munos, R. 2016. 统一计数式探索和内在动机。 在 *NIPS*。

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. 强化学习的分布式视角。 在 *ICML*。

Blundell, C.; Uria, B.; Pritzel, A.; Li, Y.; Ruderman, A.; Leibo, J. Z.; Rae, J.; Wierstra, D.; and Hassabis, D. 2016. 无模型的情景控制。 *ArXive-prints*。

Dosovitskiy, A., and Koltun, V. 2016. 通过预测未来来学习行动。 *CoRR*abs/1611.01779。

Fortunato, M.; Azar, M. G.; Piot, B.; Menick, J.; Osband, I.; Graves, A.; Mnih, V.; Munos, R.;Hassabis, D.; Pietquin, O.; Blundell, C.; and Legg, S. 2017. 噪声网络用于探索。 *CoRR*abs/1706.10295。

Hausknecht, M.,and Stone, P. 2015. 深度循环 Q- 学习用于部分可观察 MDPs。 *arXiv preprintarXiv:1507.06527*。

He, F. S.; Liu,Y.; Schwing, A. G.; and Peng, J. 2016. 一天内学会玩：通过最优性收紧实现更快的深度强化学习。 *CoRR*abs/1611.01606。

Jaderberg, M.; Mnih, V.; Czarnecki, W. M.; Schaul, T.; Leibo, J. Z.; Silver, D.; and Kavukcuoglu, K. 2016. 具有无监督辅助任务的强化学习。 *CoRR*abs/1611.05397。

Kingma, D. P., 和 Ba, J. 2014. Adam: 一种随机优化方法。在第 *3* 届国际学习表示会议（ *ICLR* ）。

Kulkarni, T. D.; Narasimhan, K.; Saeedi, A.; 和 Tenen- baum, J. B. 2016a. 分层深度强化学习：整合时间抽象和内在动机。 *CoRR*abs/1604.06057.

Kulkarni, T. D.; Saeedi, A.; Gautam, S.; 和 Gershman, S. J. 2016b. 深度后继强化学习。 *arXivpreprint arXiv:1606.02396*.

Lin, L.-J. 1992. 基于强化学习、规划和教学的自我改进反应代理。 机器学习 8(3):293–321.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; 和 Riedmiller, M. A. 2013. 使用深度强化学习玩 Atari 游戏。 *CoRR*abs/1312.5602.

Mnih, V.; Kavukcuoglu,K.;Silver,D.; Rusu,A.A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. 通过深度强化学习实现人类水平的控制。 *Nature*518(7540):529–533。

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. 深度强化学习的异步方法。 在国际机器学习会议。

Nair, A.; Srinivasan, P.; Blackwell, S.; Alcicek, C.; Fearon, R.; De Maria, A.; Panneershelvam, V.; Suleyman, M.; Beattie, C.; Petersen, S.; Legg, S.; Mnih, V.; Kavukcuoglu, K.; and Silver, D. 2015. 大规模并行深度强化学习方法。 *arXiv preprintarXiv:1507.04296*。

O' Donoghue, B.; Munos, R.; Kavukcuoglu, K.; and Mnih, V. 2016. Pgq: 结合策略梯度和 Q 学习。 *CoRR* abs/1611.01626。

Osband, I.; Blundell, C.; Pritzel, A.; and Roy, B. V. 2016. 通过引导 DQN 进行深度探索。 在 *NIPS*。

Salimans, T.; Ho, J.; Chen, X.; and Sutskever, I. 2017. 进化策略作为强化学习的可扩展替代方案。 *CoRR*abs/1703.03864。

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. 优先经验回放。 在 *Proc. of ICLR*。

Schulman, J.; Levine, S.;Moritz, P.;Jordan, M.; 和 Abbeel, P. 2015. Trust region policy optimization. 在第 *32* 届国际机器学习会议论文集 - 卷 *37*, ICML' 15, 1889– 1897. JMLR.org。

Sharma, S.; Lakshminarayanan, A. S.; 和 Ravindran, B. 2017. Learning to repeat: Finegrained action rep- etition for deep reinforcement learning. *arXiv* 预印本 *arXiv:1702.06054*。

Stadie, B. C.;Levine,S.; 和 Abbeel,P. 2015. Incentivizing exploration in reinforcement learning with deep predictive models. *CoRR*abs/1507.00814。

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. The MIT press, Cambridge MA.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine learning*3(1):9–44。

Tieleman, T., 和 Hinton, G. 2012. 讲座 6.5-rmsprop: 将梯度除以其最近幅度的运行平均值。 *COURSERA:* 机器学习的神经网络 4(2):26–31。

van Hasselt, H.; Guez, A.; Guez, A.; Hessel, M.; Mnih, V.; 和 Silver, D. 2016. 跨多个数量级的值学习。 在 神经信息处理系统进展 *29*， 4287–4295。

van Hasselt, H.; Guez, A.; 和 Silver, D. 2016. 基于双重 Q 学习的深度强化学习。 在 *AAAI* 会议录， 2094–2100。

van Hasselt, H. 2010. 双重 Q 学习。 在神经信息处理系统进展 *23*， 2613–2621。

Vezhnevets, A. S.;Osindero, S.; Schaul, T.; Heess, N.; Jader- berg, M.; Silver, D.; 和 Kavukcuoglu, K. 2017. 用于层次强化学习的联邦网络。 *CoRR*abs/1703.01161。

Wang, Z.; Schaul, T.; Hessel, M.; van Hasselt, H.; Lanctot, M.; and de Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, 1995–2003.