Continuous control with deep reinforcement learning
通过深入强化学习的持续控制

日期：2016-02-29

作者：Timothy P. Lillicrap 、Jonathan J. Hunt 、Alexander Pritzel 、Nicolas Heess 、Tom Erez 、Yuval Tassa 、David Silver 、Daan Wierstra

论文：http://arxiv.org/pdf/1509.02971v5.pdf

ABSTRACT
摘要

We adapt the ideas underlying the success of Deep Q-Learning to the continuous action domain. We present an actor-critic, model-free algorithm based on the deterministic policy gradient that can operate over continuous action spaces. Using the same learning algorithm, network architecture and hyper-parameters, our algorithm robustly solves more than 20 simulated physics tasks, including classic problems such as cartpole swing-up, dexterous manipulation, legged locomotion and car driving. Our algorithm is able to find policies whose performance is competitive with those found by a planning algorithm with full access to the dynamics of the domain and its derivatives. We further demonstrate that for many of the tasks the algorithm can learn policies “end-to-end”: directly from raw pixel inputs.

我们将Deep Q-Learning成功的基本思想应用于持续行动领域。我们提出一个基于确定性策略梯度的演员评论者，无模型算法，该算法可以在连续动作空间上运行。使用相同的学习算法，网络架构和超参数，我们的算法强大地解决了超过20个模拟物理任务，包括经典问题，例如小推车摆动，灵巧操纵，腿式运动和汽车驾驶。我们的算法能够找到策略，这些策略的性能与计划算法所发现的策略相当具有竞争力，可以充分利用域及其衍生物的动态性。我们进一步证明，对于许多任务，算法可以学习策略“端到端”：直接从原始像素输入。

1 INTRODUCTION
1引言

One of the primary goals of the field of artificial intelligence is to solve complex tasks from unprocessed, high-dimensional, sensory input. Recently, significant progress has been made by combining advances in deep learning for sensory processing (Krizhevsky et al., 2012) with reinforcement learning, resulting in the “Deep Q Network” (DQN) algorithm (Mnih et al., 2015) that is capable of human level performance on many Atari video games using unprocessed pixels for input. To do so, deep neural network function approximators were used to estimate the action-value function.

人工智能领域的主要目标之一是从未经处理的高维感官输入解决复杂任务。最近，通过将感官加工的深度学习（Krizhevsky等，2012）与强化学习相结合，取得了重大进展，形成了“Deep Q Network”（DQN）算法（Mnih等，2015）能够在许多使用未处理像素输入的Atari视频游戏上实现人性化的性能。为此，使用深度神经网络函数逼近器来估计动作值函数。

However, while DQN solves problems with high-dimensional observation spaces, it can only handle discrete and low-dimensional action spaces. Many tasks of interest, most notably physical control tasks, have continuous (real valued) and high dimensional action spaces. DQN cannot be straightforwardly applied to continuous domains since it relies on a finding the action that maximizes the action-value function, which in the continuous valued case requires an iterative optimization process at every step.

然而，尽管DQN解决了高维观测空间的问题，但它只能处理离散的和低维的动作空间。许多感兴趣的任务，最显着的是物理控制任务，具有连续（实值）和高维空间。DQN不能直接应用于连续域，因为它依赖于最大化动作值函数的动作，在连续值情况下需要在每一步都进行迭代优化。

An obvious approach to adapting deep reinforcement learning methods such as DQN to continuous domains is to to simply discretize the action space. However, this has many limitations, most notably the curse of dimensionality: the number of actions increases exponentially with the number of degrees of freedom. For example, a 7 degree of freedom system (as in the human arm) with the coarsest discretization $a_i \in \{-k, 0, k\}$ for each joint leads to an action space with dimensionality: $3^7 = 2187$. The situation is even worse for tasks that require fine control of actions as they require a correspondingly finer grained discretization, leading to an explosion of the number of discrete actions. Such large action spaces are difficult to explore efficiently, and thus successfully training DQN-like networks in this context is likely intractable. Additionally, naive discretization of action spaces needlessly throws away information about the structure of the action domain, which may be essential for solving many problems.

将深度强化学习方法（如DQN）适用于连续域的一种显而易见的方法是简单地将动作空间离散化。然而，这具有许多限制，最显着的是维度的诅咒：行动的数量随着自由度的数量呈指数增长。例如，对于每个关节，具有最粗糙离散化$a_i \in \{-k, 0, k\}$的7自由度系统（如在人类手臂中）导致具有维度的动作空间：$3^7 = 2187$。对于需要对行为进行精细控制的任务来说，情况更糟，因为这些任务需要相应的细化离散化，导致离散操作的数量激增。这样的大动作空间很难有效地探索，因此在这种情况下成功地训练类DQN网络很可能是棘手的。另外，行为空间的天真离散无用地抛弃了关于行动领域结构的信息，这对解决许多问题可能是必不可少的。

In this work we present a model-free, off-policy actor-critic algorithm using deep function approximators that can learn policies in high-dimensional, continuous action spaces. Our work is based on the deterministic policy gradient (DPG) algorithm (Silver et al., 2014) (itself similar to NFQCA (Hafner & Riedmiller, 2011), and similar ideas can be found in (Prokhorov et al., 1997)). However, as we show below, a naive application of this actor-critic method with neural function approximators is unstable for challenging problems.
在这项工作中，我们提出了一个使用深度函数逼近器的无模型，不对外演员 - 评论者算法，可以学习高维连续动作空间中的策略。我们的工作基于确定性政策梯度（DPG）算法（Silver等，2014）（本身类似于NFQCA（Hafner＆Riedmiller，2011），类似的观点可以在（Prokhorov等，1997）中找到）。然而，正如我们下面所展示的那样，这种具有神经函数逼近器的actor-critic方法的天真应用对于具有挑战性的问题是不稳定的。

⋆These authors contributed equally.
⋆这些作者均等贡献。

Here we combine the actor-critic approach with insights from the recent success of Deep Q Network (DQN) (Mnih et al., 2013; 2015). Prior to DQN, it was generally believed that learning value functions using large, non-linear function approximators was difficult and unstable. DQN is able to learn value functions using such function approximators in a stable and robust way due to two innovations: 1. the network is trained off-policy with samples from a replay buffer to minimize correlations between samples; 2. the network is trained with a target Q network to give consistent targets during temporal difference backups. In this work we make use of the same ideas, along with batch normalization (Ioffe & Szegedy, 2015), a recent advance in deep learning.
在这里，我们将演员 - 评论者的方法与最近Deep Q Network（DQN）成功的见解相结合（Mnih et al。，2013; 2015）。在DQN之前，人们普遍认为使用大型非线性函数逼近器的学习值函数是困难且不稳定的。DQN能够以稳定和强大的方式使用这些函数逼近器学习值函数，这归功于两项创新：1.网络通过来自重播缓冲区的样本进行关闭策略训练，以最小化样本之间的相关性; 2.网络接受目标Q网络的训练，在时间差异备份期间提供一致的目标。在这项工作中，我们利用了同样的想法，以及批量标准化（Ioffe＆Szegedy，2015），这是近期深度学习的进展。

In order to evaluate our method we constructed a variety of challenging physical control problems that involve complex multi-joint movements, unstable and rich contact dynamics, and gait behavior. Among these are classic problems such as the cartpole swing-up problem, as well as many new domains. A long-standing challenge of robotic control is to learn an action policy directly from raw sensory input such as video. Accordingly, we place a fixed viewpoint camera in the simulator and attempted all tasks using both low-dimensional observations (e.g. joint angles) and directly from pixels.
为了评估我们的方法，我们构建了各种具有挑战性的物理控制问题，这些问题涉及复杂的多关节运动，不稳定和丰富的接触动态以及步态行为。其中包括典型的问题，例如车轮摇摆问题，以及许多新的领域。机器人控制的长期挑战是直接从原始感官输入（如视频）中学习行动策略。因此，我们在模拟器中放置一个固定的视点相机，并尝试使用低维度观察（例如关节角度）和直接使用像素的所有任务。

Our model-free approach which we call Deep DPG (DDPG) can learn competitive policies for all of our tasks using low-dimensional observations (e.g. cartesian coordinates or joint angles) using the same hyper-parameters and network structure. In many cases, we are also able to learn good policies directly from pixels, again keeping hyperparameters and network structure constant 1.
我们称为Deep DPG（DDPG）的无模型方法可以使用相同的超参数和网络结构，使用低维观测（例如笛卡尔坐标或关节角度）学习我们所有任务的竞争策略。在很多情况下，我们还能够直接从像素中学习良好的策略，同时保持超参数和网络结构不变1。

A key feature of the approach is its simplicity: it requires only a straightforward actor-critic architecture and learning algorithm with very few “moving parts”, making it easy to implement and scale to more difficult problems and larger networks. For the physical control problems we compare our results to a baseline computed by a planner (Tassa et al., 2012) that has full access to the underlying simulated dynamics and its derivatives (see supplementary information). Interestingly, DDPG can sometimes find policies that exceed the performance of the planner, in some cases even when learning from pixels (the planner always plans over the underlying low-dimensional state space).
该方法的一个关键特征是它的简单性：它只需要一个简单的演员 - 评论者架构和学习算法，只需很少的“移动部分”，便于实施和扩展到更困难的问题和更大的网络。对于物理控制问题，我们将我们的结果与计划人员（Tassa等人，2012）计算出的基线进行比较，该基线可以完全访问底层模拟动态及其衍生物（请参阅补充信息）。有趣的是，DDPG有时可能发现超出规划者表现的策略，在某些情况下，即使从像素学习（规划者总是计划底层低维状态空间）。

2 BACKGROUND
2背景

We consider a standard reinforcement learning setup consisting of an agent interacting with an environment E in discrete timesteps. At each timestep t the agent receives an observation x_t, takes an action a_t and receives a scalar reward r_t. In all the environments considered here the actions are real-valued $a_t \in \mathbb{R}^N$. In general, the environment may be partially observed so that the entire history of the observation, action pairs $s_t = (x_1, a_1, ..., a_{t-1}, x_t)$ may be required to describe the state. Here, we assumed the environment is fully-observed so $s_t = x_t$.

我们考虑一个标准的强化学习设置，它包括一个代理以离散的时间步长与环境E进行交互。在每个时间步t，代理收到一个观察x_t，采取行动a_t并收到标量奖励r_t。在这里考虑的所有环境中，动作都是实值$a_t \in \mathbb{R}^N$。一般来说，环境可能会被部分观察到，因此观察的整个历史记录可能需要动作对$s_t = (x_1, a_1, ..., a_{t-1}, x_t)$来描述状态。在这里，我们假设环境是完全遵守的，所以$s_t = x_t$。

An agent's behavior is defined by a policy, π, which maps states to a probability distribution over the actions $\pi: \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$. The environment, E, may also be stochastic. We model it as a Markov decision process with a state space S, action space $\mathcal{A} = \mathbb{R}^N$, an initial state distribution $p(s_1)$, transition dynamics $p(s_{t+1}|s_t, a_t)$, and reward function $r(s_t, a_t)$.

代理的行为由策略π定义，该策略将状态映射到动作$\pi: \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$上的概率分布。环境E也可能是随机的。我们将它建模为具有状态空间S，动作空间$\mathcal{A} = \mathbb{R}^N$，初始状态分布$p(s_1)$，转换动力学$p(s_{t+1}|s_t, a_t)$和奖励函数$r(s_t, a_t)$的马尔可夫决策过程。

The return from a state is defined as the sum of discounted future reward $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i)$ with a discounting factor $\gamma \in [0, 1]$. Note that the return depends on the actions chosen, and therefore on the policy π, and may be stochastic. The goal in reinforcement learning is to learn a policy which maximizes the expected return from the start distribution $J = \mathbb{E}_{r_i, s_i \sim E, a_i \sim \pi}[R_1]$. We denote the discounted state visitation distribution for a policy π as ρ^π.

来自国家的收益被定义为折扣未来报酬$R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i)$与折现因子$\gamma \in [0, 1]$的总和。请注意，回报取决于所选择的行为，因此取决于策略π，并且可能是随机的。强化学习的目标是学习一项政策，使得从初始分销$J = \mathbb{E}_{r_i, s_i \sim E, a_i \sim \pi}[R_1]$的预期回报最大化。我们将政策π的折扣状态访问分布表示为ρ^π。

The action-value function is used in many reinforcement learning algorithms. It describes the expected return after taking an action a_t in state s_t and thereafter following policy π:

动作值函数用于许多强化学习算法。它描述了在a_t状态下采取行动s_t之后的预期回报，此后遵循策略π：

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r_{i \geq t}, s_{i>t} \sim E, a_{i>t} \sim \pi}[R_t | s_t, a_t] \tag{1}$$

1You can view a movie of some of the learned policies at https://goo.gl/J4PIAz

1您可以在https://goo.gl/J4PIAz查看一些学习政策的电影

Many approaches in reinforcement learning make use of the recursive relationship known as the Bellman equation:

强化学习中的许多方法利用了称为Bellman方程的递归关系：

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r(s_t, a_t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q^\pi(s_{t+1}, a_{t+1})]\right] \tag{2}$$

If the target policy is deterministic we can describe it as a function μ : S ← A and avoid the inner expectation:

$Q^\mu(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu(s_{t+1}))]$ (3)

如果目标政策是确定性的，我们可以将其描述为一个函数μ：S←A并避免内部期望：$Q^\mu(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu(s_{t+1}))]$ (3)

The expectation depends only on the environment. This means that it is possible to learn Q^μ offpolicy, using transitions which are generated from a different stochastic behavior policy β.

期望仅取决于环境。这意味着可以学习Q^μ offpolicy，使用从不同的随机行为策略β生成的转换。

Q-learning (Watkins & Dayan, 1992), a commonly used off-policy algorithm, uses the greedy policy $\mu(s) = \arg\max_a Q(s, a)$. We consider function approximators parameterized by θ^Q, which we optimize by minimizing the loss:

Q-learning（Watkins＆Dayan，1992）是一种常用的关闭策略算法，它使用贪婪策略$\mu(s) = \arg\max_a Q(s, a)$。我们考虑由$\theta^Q$进行参数化的函数逼近器，我们通过最小化损失来进行优化：

$$L(\theta^Q) = \mathbb{E}_{s_t \sim \rho^\beta, a_t \sim \beta, r_t \sim E}\left[\left(Q(s_t, a_t|\theta^Q) - y_t\right)^2\right] \tag{4}$$

where

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta^Q). \tag{5}$$

While y_t is also dependent on θ^Q, this is typically ignored.
虽然y_t也依赖于θ^Q，但这通常被忽略。

The use of large, non-linear function approximators for learning value or action-value functions has often been avoided in the past since theoretical performance guarantees are impossible, and practically learning tends to be unstable. Recently, (Mnih et al., 2013; 2015) adapted the Q-learning algorithm in order to make effective use of large neural networks as function approximators. Their algorithm was able to learn to play Atari games from pixels. In order to scale Q-learning they introduced two major changes: the use of a replay buffer, and a separate target network for calculating y_t. We employ these in the context of DDPG and explain their implementation in the next section.
过去通常避免使用大型的非线性函数逼近器来学习价值或行为 - 价值函数，因为理论性能保证是不可能的，并且实际上学习趋于不稳定。最近，（Mnih等，2013; 2015）改编了Q学习算法，以便有效地使用大型神经网络作为函数逼近器。他们的算法能够学习如何使用像素来玩Atari游戏。为了扩展Q学习，他们引入了两个主要的改变：使用重播缓冲区和用于计算y_t的单独目标网络。我们在DDPG的背景下使用这些内容，并在下一部分解释它们的实现。

3 ALGORITHM
3算法

It is not possible to straightforwardly apply Q-learning to continuous action spaces, because in continuous spaces finding the greedy policy requires an optimization of a_t at every timestep; this optimization is too slow to be practical with large, unconstrained function approximators and nontrivial action spaces. Instead, here we used an actor-critic approach based on the DPG algorithm (Silver et al., 2014).
将Q学习直接应用于连续动作空间是不可能的，因为在连续空间中发现贪婪策略需要在每个时间步都优化a_t;这种优化速度太慢，不适用于大的，无约束的函数逼近器和非平凡的动作空间。相反，在这里我们使用了基于DPG算法的演员 - 评论者方法（Silver et al。，2014）。

The DPG algorithm maintains a parameterized actor function $\mu(s|\theta^\mu)$ which specifies the current policy by deterministically mapping states to a specific action. The critic $Q(s,a)$ is learned using the Bellman equation as in Q-learning. The actor is updated by following the applying the chain rule to the expected return from the start distribution J with respect to the actor parameters:
DPG算法维护一个参数化的参数化函数$\mu(s|\theta^\mu)$，它通过将状态确定性地映射到特定操作来指定当前策略。批评者$Q(s,a)$是在Q-learning中使用Bellman方程学习的。通过将链规则应用于从起始分布J相对于参数参数的预期回报来更新参与者：

$$\begin{aligned}\nabla_{\theta^\mu} J &\approx \mathbb{E}_{s_t\sim\rho^\beta}\left[\nabla_{\theta^\mu} Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t|\theta^\mu)}\right] \\ &= \mathbb{E}_{s_t\sim\rho^\beta}\left[\nabla_a Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t)}\nabla_{\theta_\mu}\mu(s|\theta^\mu)|_{s=s_t}\right]\end{aligned} \tag{6}$$

Silver et al. (2014) proved that this is the policy gradient, the gradient of the policy's performance 2.
Silver等人。（2014）证明，这是政策梯度，政策表现的梯度2。

As with Q learning, introducing non-linear function approximators means that convergence is no longer guaranteed. However, such approximators appear essential in order to learn and generalize on large state spaces. NFQCA (Hafner & Riedmiller, 2011), which uses the same update rules as DPG but with neural network function approximators, uses batch learning for stability, which is intractable for large networks. A minibatch version of NFQCA which does not reset the policy at each update, as would be required to scale to large networks, is equivalent to the original DPG, which we compare to here. Our contribution here is to provide modifications to DPG, inspired by the success of DQN, which allow it to use neural network function approximators to learn in large state and action spaces online. We refer to our algorithm as Deep DPG (DDPG, Algorithm 1).
与Q学习一样，引入非线性函数逼近器意味着不再保证收敛。然而，为了学习和推广大型状态空间，这样的近似者看起来是必不可少的。NFQCA（Hafner＆Riedmiller，2011）使用与DPG相同的更新规则，但是使用神经网络功能逼近器，使用批量学习来保持稳定性，这对于大型网络来说是棘手的。NFQCA的minibatch版本不会在每次更新时重置策略，这将需要扩展到大型网络，这与我们在此比较的原始DPG相同。我们的贡献是为DPG提供修改，这受到DQN的成功启发，它允许它使用神经网络函数逼近器在线状态和动作空间中学习。我们将我们的算法称为Deep DPG（DDPG，算法1）。

2In practice, as in commonly done in policy gradient implementations, we ignored the discount in the statevisitation distribution ρ^β.
2在实践中，正如在政策梯度实施中通常所做的那样，我们忽略了状态访问分布ρ^β中的折扣。

One challenge when using neural networks for reinforcement learning is that most optimization algorithms assume that the samples are independently and identically distributed. Obviously, when the samples are generated from exploring sequentially in an environment this assumption no longer holds. Additionally, to make efficient use of hardware optimizations, it is essential to learn in minibatches, rather than online.
当使用神经网络进行强化学习时，一个挑战是大多数优化算法都假设样本是独立且相同分布的。显然，当样本是在环境中依次探索而生成时，这个假设不再成立。此外，为了有效利用硬件优化，学习minibatches而不是在线学习是非常重要的。

As in DQN, we used a replay buffer to address these issues. The replay buffer is a finite sized cache R. Transitions were sampled from the environment according to the exploration policy and the tuple (s_t, a_t, r_t, s_{t+1}) was stored in the replay buffer. When the replay buffer was full the oldest samples were discarded. At each timestep the actor and critic are updated by sampling a minibatch uniformly from the buffer. Because DDPG is an off-policy algorithm, the replay buffer can be large, allowing the algorithm to benefit from learning across a set of uncorrelated transitions.
和DQN一样，我们使用重播缓冲区来解决这些问题。重放缓冲区是一个有限大小的缓存R.根据探索策略从环境中采样转换，并将元组(s_t, a_t, r_t, s_{t+1})存储在重放缓冲区中。当重播缓冲区已满时，丢弃最旧的样本。在每个时间步，演员和评论者通过从缓冲器统一取样小批次来更新。由于DDPG是一种关闭策略算法，因此重播缓冲区可能较大，从而允许通过一组不相关的转换学习算法。

Directly implementing Q learning (equation 4) with neural networks proved to be unstable in many environments. Since the network $Q(s, a|\theta^Q)$ being updated is also used in calculating the target value (equation 5), the Q update is prone to divergence. Our solution is similar to the target network used in (Mnih et al., 2013) but modified for actor-critic and using “soft” target updates, rather than directly copying the weights. We create a copy of the actor and critic networks, $Q'(s, a|\theta^{Q'})$ and $\mu'(s|\theta^{\mu'})$ respectively, that are used for calculating the target values. The weights of these target networks are then updated by having them slowly track the learned networks: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ with $\tau \ll 1$. This means that the target values are constrained to change slowly, greatly improving the stability of learning. This simple change moves the relatively unstable problem of learning the action-value function closer to the case of supervised learning, a problem for which robust solutions exist. We found that having both a target μ' and Q' was required to have stable targets y_i in order to consistently train the critic without divergence. This may slow learning, since the target network delays the propagation of value estimations. However, in practice we found this was greatly outweighed by the stability of learning.
用神经网络直接实现Q学习（方程4）在许多环境中被证明是不稳定的。由于正在更新的网络$Q(s, a|\theta^Q)$也用于计算目标值（等式5），所以Q更新容易发散。我们的解决方案与（Mnih et al。，2013）中使用的目标网络类似，但是对演员评论者进行修改并使用“软”目标更新，而不是直接复制权重。我们分别创建了用于计算目标值的演员和评论者网络副本$Q'(s, a|\theta^{Q'})$和$\mu'(s|\theta^{\mu'})$。然后通过让这些目标网络慢慢跟踪学习网络来更新这些目标网络的权重：带$\tau \ll 1$的$\theta' \leftarrow \tau\theta + (1-\tau)\theta'$。这意味着目标值被限制缓慢变化，极大地提高了学习的稳定性。这种简单的变化将相对不稳定的动作值函数学习问题更接近监督学习的情况，这是一个存在鲁棒解的问题。我们发现同时拥有目标μ'和Q'需要有稳定的目标y_i才能始终如一地训练评论者而不会发散。这可能会减慢学习速度，因为目标网络会延迟价值估计的传播。但是，实际上我们发现这一点远远超过了学习的稳定性。

When learning from low dimensional feature vector observations, the different components of the observation may have different physical units (for example, positions versus velocities) and the ranges may vary across environments. This can make it difficult for the network to learn effectively and may make it difficult to find hyper-parameters which generalise across environments with different scales of state values.
当从低维特征向量观测中学习时，观测的不同组分可能具有不同的物理单位（例如，位置与速度），并且范围可能在不同的环境中变化。这可能会使网络很难有效地学习，并且很难找到在具有不同的状态值范围的环境中泛化的超参数。

One approach to this problem is to manually scale the features so they are in similar ranges across environments and units. We address this issue by adapting a recent technique from deep learning called batch normalization (Ioffe & Szegedy, 2015). This technique normalizes each dimension across the samples in a minibatch to have unit mean and variance. In addition, it maintains a running average of the mean and variance to use for normalization during testing (in our case, during exploration or evaluation). In deep networks, it is used to minimize covariance shift during training, by ensuring that each layer receives whitened input. In the low-dimensional case, we used batch normalization on the state input and all layers of the μ network and all layers of the Q network prior to the action input (details of the networks are given in the supplementary material). With batch normalization, we were able to learn effectively across many different tasks with differing types of units, without needing to manually ensure the units were within a set range.
解决此问题的一种方法是手动缩放功能，以使它们跨环境和单位处于类似范围内。我们通过调整最近的一项名为批量标准化的深度学习技术来解决这个问题（Ioffe＆Szegedy，2015）。该技术对小批量样本中的每个维度进行归一化，以得到单位均值和方差。另外，它在测试期间（在我们的情况下，在勘探或评估期间）保持均值和方差的运行平均值用于标准化。在深度网络中，通过确保每个层接收到白化输入，它被用于最小化训练期间的协方差偏移。在低维情况下，我们在动作输入之前在状态输入和μ网络的所有层以及Q网络的所有层上使用了批量标准化（网络细节在补充材料中给出）。通过批量标准化，我们能够跨不同类型的单元有效地学习许多不同的任务，而无需手动确保单元在设定的范围内。

A major challenge of learning in continuous action spaces is exploration. An advantage of offpolicies algorithms such as DDPG is that we can treat the problem of exploration independently from the learning algorithm. We constructed an exploration policy μ' by adding noise sampled from a noise process N to our actor policy $\mu'(s_t) = \mu(s_t|\theta_t^\mu) + \mathcal{N}$ (7) N can be chosen to chosen to suit the environment. As detailed in the supplementary materials we used an Ornstein-Uhlenbeck process (Uhlenbeck & Ornstein, 1930) to generate temporally correlated exploration for exploration efficiency in physical control problems with inertia (similar use of autocorrelated noise was introduced in (Wawrzy´nski, 2015)).
在连续行动空间学习的一个主要挑战是探索。像DDPG这样的失衡策略算法的一个优点是我们可以独立于学习算法来处理探索问题。我们通过将我们的演员政策$\mu'(s_t) = \mu(s_t|\theta_t^\mu) + \mathcal{N}$（7）中的噪音过程N中的噪音加入来构建探索政策$\mu'$（7）可以选择N来适应环境。如补充材料所详述的，我们使用Ornstein-Uhlenbeck过程（Uhlenbeck＆Ornstein，1930）为具有惯性的物理控制问题中的勘探效率产生时间相关勘探（Wawrzy'nski，2015）中引入了自相关噪声的类似用法，）。

4 RESULTS
4结果

We constructed simulated physical environments of varying levels of difficulty to test our algorithm. This included classic reinforcement learning environments such as cartpole, as well as difficult, high dimensional tasks such as gripper, tasks involving contacts such as puck striking (canada) and locomotion tasks such as cheetah (Wawrzy´nski, 2009). In all domains but cheetah the actions were torques applied to the actuated joints. These environments were simulated using MuJoCo (Todorov et al., 2012). Figure 1 shows renderings of some of the environments used in the task (the supplementary contains details of the environments and you can view some of the learned policies at https://goo.gl/J4PIAz).
我们构建了不同难度等级的模拟物理环境来测试我们的算法。这包括经典的强化学习环境，例如cartpole，以及难度较大的高维度任务，如抓手，涉及诸如击球（加拿大）等接触的任务以及猎豹（Wawrzy'nski，2009）等运动任务。在所有领域，但猎豹的行动扭矩施加到驱动关节。使用MuJoCo模拟这些环境（Todorov等，2012）。图1显示了任务中使用的一些环境的渲染（补充内容包含环境的详细信息，您可以通过https://goo.gl/J4PIAz查看一些学习到的策略）。

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

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

  end for
end for

In all tasks, we ran experiments using both a low-dimensional state description (such as joint angles and positions) and high-dimensional renderings of the environment. As in DQN (Mnih et al., 2013; 2015), in order to make the problems approximately fully observable in the high dimensional environment we used action repeats. For each timestep of the agent, we step the simulation 3 timesteps, repeating the agent's action and rendering each time. Thus the observation reported to the agent contains 9 feature maps (the RGB of each of the 3 renderings) which allows the agent to infer velocities using the differences between frames. The frames were downsampled to 64x64 pixels and the 8-bit RGB values were converted to floating point scaled to $|0, 1|$. See supplementary information for details of our network structure and hyperparameters.
在所有任务中，我们都使用低维状态描述（例如关节角度和位置）以及环境的高维渲染来进行实验。和DQN一样（Mnih et al。，2013; 2015），为了使这些问题在高维环境中几乎可以完全观察到，我们使用动作重复。对于代理的每个时间步，我们进行模拟3次步骤，每次重复代理的动作和渲染。因此，向代理报告的观察结果包含9个特征地图（3个渲染中的每一个的RGB），这使得代理可以使用帧之间的差异来推断速度。帧被下采样到64x64像素，并且8位RGB值被转换为缩放到$|0, 1|$的浮点。有关我们网络结构和超参数的详细信息，请参阅补充信息。

We evaluated the policy periodically during training by testing it without exploration noise. Figure 2 shows the performance curve for a selection of environments. We also report results with components of our algorithm (i.e. the target network or batch normalization) removed. In order to perform well across all tasks, both of these additions are necessary. In particular, learning without a target network, as in the original work with DPG, is very poor in many environments.

我们在培训期间定期评估政策，通过在没有探索噪音的情况下测试它图2显示了选择环境的性能曲线。我们还报告了我们算法的组件（即目标网络或批量标准化）的结果。为了在所有任务中表现良好，这两项增加都是必要的。特别是，在没有目标网络的情况下学习，就像在DPG的原始工作中一样，在许多环境中都很差。

Surprisingly, in some simpler tasks, learning policies from pixels is just as fast as learning using the low-dimensional state descriptor. This may be due to the action repeats making the problem simpler. It may also be that the convolutional layers provide an easily separable representation of state space, which is straightforward for the higher layers to learn on quickly.
令人惊讶的是，在一些更简单的任务中，像素学习策略与使用低维状态描述符学习一样快。这可能是由于行动重复使问题变得更简单。它也可能是卷积层提供了一个易于分离的状态空间表示，这对于更高层快速学习来说是直接的。

Table 1 summarizes DDPG's performance across all of the environments (results are averaged over 5 replicas). We normalized the scores using two baselines. The first baseline is the mean return from a naive policy which samples actions from a uniform distribution over the valid action space. The second baseline is iLQG (Todorov & Li, 2005), a planning based solver with full access to the underlying physical model and its derivatives. We normalize scores so that the naive policy has a mean score of 0 and iLQG has a mean score of 1. DDPG is able to learn good policies on many of the tasks, and in many cases some of the replicas learn policies which are superior to those found by iLQG, even when learning directly from pixels.
表1总结了DDPG在所有环境中的表现（结果平均为5次复制）。我们使用两个基线将分数归一化。第一个基线是来自天真政策的平均回报，该政策从有效行动空间的均匀分布中采取行动。第二个基准是iLQG（Todorov＆Li，2005），这是一个基于规划的求解器，可以完全访问基础物理模型及其衍生物。我们对分数进行归一化处理，以使幼稚政策的平均得分为0，iLQG的平均得分为1。DDPG能够针对许多任务学习良好的策略，并且在许多情况下，一些副本学习优于iLQG所发现策略的策略，即使在直接从像素中学习时也是如此。

It can be challenging to learn accurate value estimates. Q-learning, for example, is prone to overestimating values (Hasselt, 2010). We examined DDPG's estimates empirically by comparing the values estimated by Q after training with the true returns seen on test episodes. Figure 3 shows that in simple tasks DDPG estimates returns accurately without systematic biases. For harder tasks the Q estimates are worse, but DDPG is still able learn good policies.
学习准确的价值估计可能是一项挑战。例如，Q-learning容易高估价值（Hasselt，2010）。我们通过比较训练后Q值和测试期间看到的真实回报，凭经验检验了DDPG的估计值。图3显示，在简单任务中，DDPG可以准确估算收益，而无需系统偏见。对于较难的任务，Q估计会更糟，但DDPG仍然能够学习良好的政策。

To demonstrate the generality of our approach we also include Torcs, a racing game where the actions are acceleration, braking and steering. Torcs has previously been used as a testbed in other policy learning approaches (Koutn´ık et al., 2014b). We used an identical network architecture and learning algorithm hyper-parameters to the physics tasks but altered the noise process for exploration because of the very different time scales involved. On both low-dimensional and from pixels, some replicas were able to learn reasonable policies that are able to complete a circuit around the track though other replicas failed to learn a sensible policy.
为了证明我们的方法的一般性，我们还包括Torcs，一款赛车游戏，其中的动作是加速，制动和转向。Torcs之前曾被用作其他政策学习方法的试验平台（Koutn'ıket al。，2014b）。我们对物理任务使用了相同的网络结构和学习算法超参数，但由于所涉及的时间尺度不同，因此改变了探测的噪声过程。在低维和像素上，一些复制品能够学习合理的策略，能够在轨道周围完成电路，但其他复制品未能学习明智的策略。

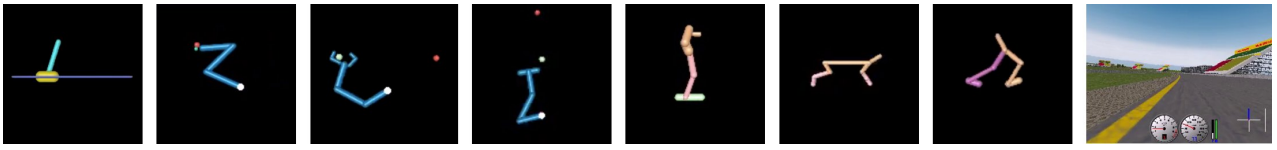
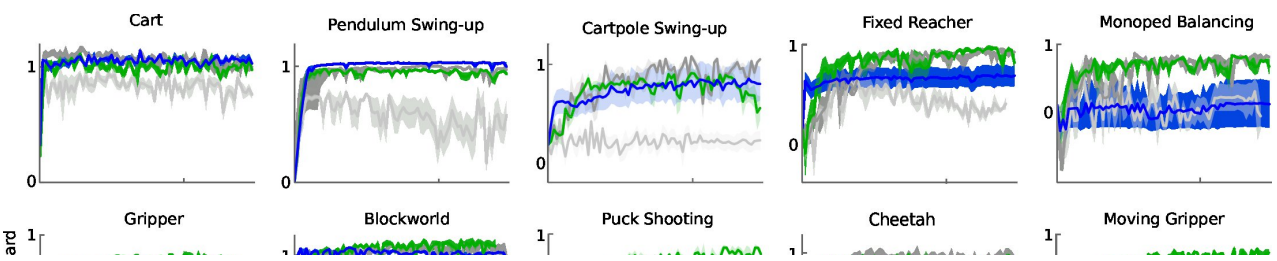


Figure 1: Example screenshots of a sample of environments we attempt to solve with DDPG. In order from the left: the cartpole swing-up task, a reaching task, a gasp and move task, a puck-hitting task, a monoped balancing task, two locomotion tasks and Torcs (driving simulator). We tackle all tasks using both low-dimensional feature vector and high-dimensional pixel inputs. Detailed descriptions of the environments are provided in the supplementary. Movies of some of the learned policies are available at https://goo.gl/J4PIAz.
图1：我们尝试使用DDPG解决的环境示例屏幕截图。从左侧开始：小推杆摆动任务，伸手任务，喘气和移动任务，击球任务，单人平衡任务，两个运动任务和Torcs（驾驶模拟器）。我们使用低维特征向量和高维像素输入来处理所有任务。补充文件提供了有关环境的详细说明。https://goo.gl/J4PIAz上提供了部分学习政策的电影。



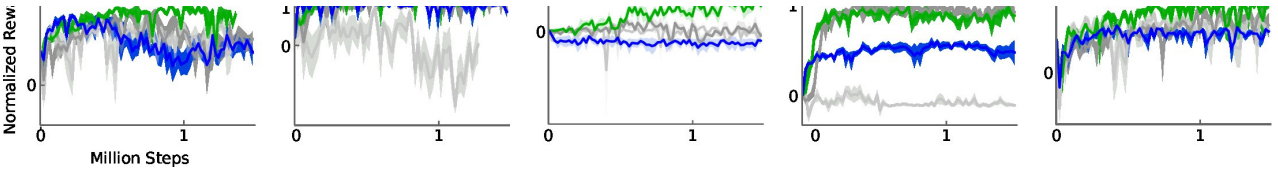


Figure 2: Performance curves for a selection of domains using variants of DPG: original DPG algorithm (minibatch NFQCA) with batch normalization (light grey), with target network (dark grey), with target networks and batch normalization (green), with target networks from pixel-only inputs (blue). Target networks are crucial.

图2：使用DPG变体的选择域的性能曲线：具有批量标准化（浅灰色），具有目标网络（深灰色），具有目标网络和批量标准化（绿色）的原始DPG算法（小批次NFQCA）网络来自仅有像素的输入（蓝色）。目标网络至关重要。

5 RELATED WORK
5相关工作

The original DPG paper evaluated the algorithm with toy problems using tile-coding and linear function approximators. It demonstrated data efficiency advantages for off-policy DPG over both on- and off-policy stochastic actor critic. It also solved one more challenging task in which a multijointed octopus arm had to strike a target with any part of the limb. However, that paper did not demonstrate scaling the approach to large, high-dimensional observation spaces as we have here.

最初的DPG论文使用平铺编码和线性函数逼近器评估具有玩具问题的算法。它证明了在策略外DPG方面数据效率优于策略和随机角色评论者的优势。它还解决了一个更具挑战性的任务，其中一个多关节章鱼不得不用肢体的任何部分击中目标。但是，该论文并没有证明我们在这里已经提出了扩展大型高维观测空间的方法。

It has often been assumed that standard policy search methods such as those explored in the present work are simply too fragile to scale to difficult problems (Levine et al., 2015). Standard policy search is thought to be difficult because it deals simultaneously with complex environmental dynamics and a complex policy. Indeed, most past work with actor-critic and policy optimization approaches have had difficulty scaling up to more challenging problems (Deisenroth et al., 2013). Typically, this is due to instability in learning wherein progress on a problem is either destroyed by subsequent learning updates, or else learning is too slow to be practical.

人们经常认为，标准的政策搜索方法，例如目前工作中探索的方法，太脆弱了，难以扩展到困难问题（Levine et al。，2015）。标准的政策搜索被认为是困难的，因为它同时处理复杂的环境动态和复杂的政策。事实上，大多数过去与演员批评和政策优化方法的工作都难以扩展到更具挑战性的问题（Deisenroth et al。，2013）。通常，这是由于学习的不稳定性，其中问题的进展要么被随后的学习更新所破坏，要么学习过于缓慢而不切实际。

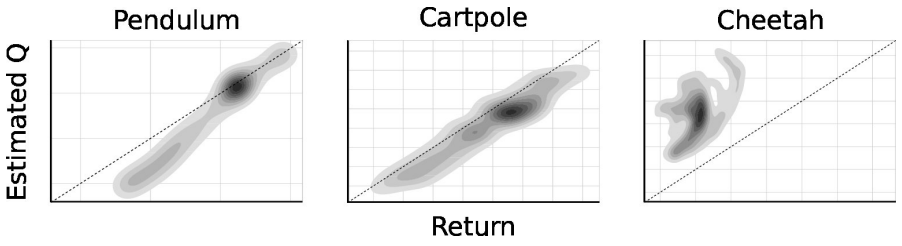


Figure 3: Density plot showing estimated Q values versus observed returns sampled from test episodes on 5 replicas. In simple domains such as pendulum and cartpole the Q values are quite accurate. In more complex tasks, the Q estimates are less accurate, but can still be used to learn competent policies. Dotted line indicates unity, units are arbitrary.

图3：密度图，显示估计的Q值与从5个副本上的测试剧集中抽取的观察到的收益率。在简单的领域，例如钟摆和Cartpole，Q值相当准确。在更复杂的任务中，Q估算不太准确，但仍可用于学习合格的策略。虚线表示统一，单位是任意的。

Table 1: Performance after training across all environments for at most 2.5 million steps. We report both the average and best observed (across 5 runs). All scores, except Torcs, are normalized so that a random agent receives 0 and a planning algorithm 1; for Torcs we present the raw reward score. We include results from the DDPG algorithn in the low-dimensional (lowd) version of the environment and high-dimensional (pix). For comparision we also include results from the original DPG algorithm with a replay buffer and batch normalization (cntrl).

表1：在所有环境中进行最多250万步训练后的表现。我们报告平均值和最佳观察值（5次运行）。除Torcs以外的所有得分都被标准化，以便随机代理接收0和计划算法1;对于Torcs，我们展示原始奖励分数。我们在低维（lowd）版本的环境和高维（pix）中包含DDPG算法的结果。为了比较，我们还包括原始DPG算法的结果和重放缓冲区以及批量归一化（cntrl）。

environment	$R_{av,lowd}$	$R_{best,lowd}$	$R_{av,pix}$	$R_{best,pix}$	$R_{av,cntrl}$	$R_{best,cntrl}$
blockworld1	1.156	1.511	0.466	1.299	-0.080	1.260

blockworld3da	0.340	0.705	0.889	2.225	-0.139	0.658
canada	0.303	1.735	0.176	0.688	0.125	1.157
canada2d	0.400	0.978	-0.285	0.119	-0.045	0.701
cart	0.938	1.336	1.096	1.258	0.343	1.216
cartpole	0.844	1.115	0.482	1.138	0.244	0.755
cartpoleBalance	0.951	1.000	0.335	0.996	-0.468	0.528
cartpoleParallelDouble	0.549	0.900	0.188	0.323	0.197	0.572
cartpoleSerialDouble	0.272	0.719	0.195	0.642	0.143	0.701
cartpoleSerialTriple	0.736	0.946	0.412	0.427	0.583	0.942
cheetah	0.903	1.206	0.457	0.792	-0.008	0.425
fixedReacher	0.849	1.021	0.693	0.981	0.259	0.927
fixedReacherDouble	0.924	0.996	0.872	0.943	0.290	0.995
fixedReacherSingle	0.954	1.000	0.827	0.995	0.620	0.999
gripper	0.655	0.972	0.406	0.790	0.461	0.816
gripperRandom	0.618	0.937	0.082	0.791	0.557	0.808
hardCheetah	1.311	1.990	1.204	1.431	-0.031	1.411
hopper	0.676	0.936	0.112	0.924	0.078	0.917
hyq	0.416	0.722	0.234	0.672	0.198	0.618
movingGripper	0.474	0.936	0.480	0.644	0.416	0.805
pendulum	0.946	1.021	0.663	1.055	0.099	0.951
reacher	0.720	0.987	0.194	0.878	0.231	0.953
reacher3daFixedTarget	0.585	0.943	0.453	0.922	0.204	0.631
reacher3daRandomTarget	0.467	0.739	0.374	0.735	-0.046	0.158
reacherSingle	0.981	1.102	1.000	1.083	1.010	1.083
walker2d	0.705	1.573	0.944	1.476	0.393	1.397
torcs	-393.385	1840.036	-401.911	1876.284	-911.034	1961.600

Recent work with model-free policy search has demonstrated that it may not be as fragile as previously supposed. Wawrzy´nski (2009); Wawrzy´nski & Tanwani (2013) has trained stochastic policies in an actor-critic framework with a replay buffer. Concurrent with our work, Balduzzi & Ghifary (2015) extended the DPG algorithm with a "deviator" network which explicitly learns $\partial Q/\partial a$. However, they only train on two low-dimensional domains. Heess et al. (2015) introduced SVG(0) which also uses a Q-critic but learns a stochastic policy. DPG can be considered the deterministic limit of SVG(0). The techniques we described here for scaling DPG are also applicable to stochastic policies by using the reparametrization trick (Heess et al., 2015; Schulman et al., 2015a).

最近与无模型政策搜索的工作表明，它可能不像以前假设的那样脆弱。Wawrzy'nski（2009）; Wawrzy'nski和Tanwani（2013）已经在带有重播缓冲区的演员评论框架中训练了随机策略。与我们的工作同时，Balduzzi＆Ghifary（2015）将DPG算法扩展为明确学习$\partial Q/\partial a$的"偏差"网络。但是，他们只训练两个低维领域。Heess等人。（2015）引入了SVG（0），它也使用了Q-critic，但学习了一个随机策略。DPG可以被认为是SVG（0）的确定性极限。我们在这里描述的用于缩放DPG的技术也适用于通过使用reparametrization技巧的随机策略（Heess等，2015; Schulman等，2015a）。

Another approach, trust region policy optimization (TRPO) (Schulman et al., 2015b), directly constructs stochastic neural network policies without decomposing problems into optimal control and supervised phases. This method produces near monotonic improvements in return by making carefully chosen updates to the policy parameters, constraining updates to prevent the new policy from diverging too far from the existing policy. This approach does not require learning an action-value function, and (perhaps as a result) appears to be significantly less data efficient.

另一种方法，即信赖域策略优化（TRPO）（Schulman等，2015b）直接构造随机神经网络策略，而不将问题分解为最优控制和监督阶段。这种方法通过仔细选择更新策略参数来产生近乎单调的改进，从而限制更新以防止新策略偏离现有策略。这种方法不需要学习一个动作值函数，并且（可能作为结果）似乎显着减少了数据的有效性。

To combat the challenges of the actor-critic approach, recent work with guided policy search (GPS) algorithms (e.g., (Levine et al., 2015)) decomposes the problem into three phases that are relatively easy to solve: first, it uses full-state observations to create locally-linear approximations of the dynamics around one or more nominal trajectories, and then uses optimal control to find the locally-linear optimal policy along these trajectories; finally, it uses supervised learning to train a complex, non-linear policy (e.g. a deep neural network) to reproduce the state-to-action mapping of the optimized trajectories.

为了应对行为者 - 评论者方法的挑战，最近与指导性策略搜索（GPS）算法（例如（Levine等，2015））合作将问题分解为三个相对容易解决的问题：首先，它使用全态观测以创建围绕一个或多个标称轨迹的动态局部线性近似，然后使用最优控制在这些轨迹上找出局部线性最优策略;最后，它使用监督式学习来训练复杂的非线性策略（例如深度神经网络）以重现优化轨迹的状态到行为的映射。

This approach has several benefits, including data efficiency, and has been applied successfully to a variety of real-world robotic manipulation tasks using vision. In these tasks GPS uses a similar convolutional policy network to ours with 2 notable exceptions: 1. it uses a spatial softmax to reduce the dimensionality of visual features into a single (x, y) coordinate for each feature map, and 2. the policy also receives direct low-dimensional state information about the configuration of the robot at the first fully connected layer in the network. Both likely increase the power and data efficiency of the algorithm and could easily be exploited within the DDPG framework.
这种方法有几个好处，包括数据效率，并已成功应用于各种使用视觉的真实世界机器人操纵任务。在这些任务中，GPS使用与我们类似的卷积策略网络，其中有两个值得注意的例外：1.它使用空间softmax将视觉特征的维度减少为每个特征地图的单个(x, y)坐标，2.该策略还接收直接低有关机器人在网络中第一个完全连接层上的配置的三维状态信息。两者都可能增加算法的功效和数据效率，并且可以在DDPG框架内轻松利用。

PILCO (Deisenroth & Rasmussen, 2011) uses Gaussian processes to learn a non-parametric, probabilistic model of the dynamics. Using this learned model, PILCO calculates analytic policy gradients and achieves impressive data efficiency in a number of control problems. However, due to the high computational demand, PILCO is “impractical for high-dimensional problems” (Wahlstr¨om et al., 2015). It seems that deep function approximators are the most promising approach for scaling reinforcement learning to large, high-dimensional domains.
PILCO（Deisenroth＆Rasmussen，2011）使用高斯过程来学习动力学的非参数，概率模型。使用这种学习模型，PILCO可以计算分析政策梯度，并在许多控制问题中实现令人印象深刻的数据效率。然而，由于高计算需求，PILCO对于高维问题是不切实际的（Wahlstr¨omet al。，2015）。似乎深度函数逼近器是将强化学习扩展到大型高维域的最有前途的方法。

Wahlstr¨om et al. (2015) used a deep dynamical model network along with model predictive control to solve the pendulum swing-up task from pixel input. They trained a differentiable forward model and encoded the goal state into the learned latent space. They use model-predictive control over the learned model to find a policy for reaching the target. However, this approach is only applicable to domains with goal states that can be demonstrated to the algorithm.
Wahlströom等人（2015）使用了一个深度动力学模型网络和模型预测控制来解决像素输入的摆动摆动任务。他们训练了一个可区分的前向模型，并将目标状态编码到学习的潜在空间中。他们使用对学习模型的模型预测控制来找出实现目标的策略。但是，这种方法仅适用于可以向演算法演示的具有目标状态的域。

Recently, evolutionary approaches have been used to learn competitive policies for Torcs from pixels using compressed weight parametrizations (Koutn´ık et al., 2014a) or unsupervised learning (Koutn´ık et al., 2014b) to reduce the dimensionality of the evolved weights. It is unclear how well these approaches generalize to other problems.
最近，已经使用演化方法来学习使用压缩权重参数化（Koutn'ık等，2014a）或无监督学习（Koutn'ık等，2014b）从像素到Torcs的竞争策略，以降低演化权重的维数。目前还不清楚这些方法对其他问题有多普遍。

6 CONCLUSION
六，结论

The work combines insights from recent advances in deep learning and reinforcement learning, resulting in an algorithm that robustly solves challenging problems across a variety of domains with continuous action spaces, even when using raw pixels for observations. As with most reinforcement learning algorithms, the use of non-linear function approximators nullifies any convergence guarantees; however, our experimental results demonstrate that stable learning without the need for any modifications between environments.
该工作结合了深入学习和强化学习最新进展的见解，从而产生一种算法，即使在使用原始像素进行观察时，该算法也可以稳健地解决具有连续动作空间的各种领域中具有挑战性的问题。与大多数强化学习算法一样，使用非线性函数逼近器也无法保证任何收敛性。然而，我们的实验结果表明，稳定的学习不需要任何环境之间的修改。

Interestingly, all of our experiments used substantially fewer steps of experience than was used by DQN learning to find solutions in the Atari domain. Nearly all of the problems we looked at were solved within 2.5 million steps of experience (and usually far fewer), a factor of 20 fewer steps than DQN requires for good Atari solutions. This suggests that, given more simulation time, DDPG may solve even more difficult problems than those considered here.
有趣的是，我们所有的实验都比DQN学习在Atari域中找到解决方案所用的经验步骤少得多。几乎所有我们看到的问题都在250万步的体验（通常少得多）内解决，这比DQN需要的步骤要少20个步骤，以获得良好的Atari解决方案。这表明，考虑到更多的模拟时间，DDPG可能会解决比这里考虑的更困难的问题。

A few limitations to our approach remain. Most notably, as with most model-free reinforcement approaches, DDPG requires a large number of training episodes to find solutions. However, we believe that a robust model-free approach may be an important component of larger systems which may attack these limitations (Gl¨ascher et al., 2010).
我们的方法仍然存在一些限制。最值得注意的是，与大多数无模型强化方法一样，DDPG需要大量的训练集才能找到解决方案。然而，我们认为一个强大的无模型方法可能是可能攻击这些限制的较大系统的一个重要组成部分（Gl¨ascheret al。，2010）。

REFERENCES

参考

Balduzzi, David and Ghifary, Muhammad. Compatible value gradients for reinforcement learning of continuous deep policies. arXiv preprint arXiv:1509.03005, 2015.

Balduzzi，David和Ghifary，穆罕默德。兼容的价值梯度强化学习连续的深层政策。 arXiv预印本arXiv：1509.03005,2015。

Deisenroth, Marc and Rasmussen, Carl E. Pilco: A model-based and data-efficient approach to policy search. In Proceedings of the 28th International Conference on machine learning (ICML11), pp. 465–472, 2011.

Deisenroth，Marc和Rasmussen，Carl E. Pilco：基于模型和数据有效的政策搜索方法。在第28届国际机器学习会议论文集（ICML11），第465-472页，2011年。

Deisenroth, Marc Peter, Neumann, Gerhard, Peters, Jan, et al. A survey on policy search for robotics. Foundations and Trends in Robotics, 2(1-2):1–142, 2013.

Deisenroth，Marc Peter，Neumann，Gerhard，Peters，Jan等人。机器人技术政策研究综述。基础和机器人趋势，2（1-2）：1-142，2013。

Gl¨ascher, Jan, Daw, Nathaniel, Dayan, Peter, and O'Doherty, John P. States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. Neuron, 66(4):585–595, 2010.

Gl¨ascher，Jan，Daw，Nathaniel，Dayan，Peter和O'Doherty，John P.状态与奖励：基于模型和无模型强化学习的可分离神经预测误差信号。Neuron，66（4）：585-595，2010。

Glorot, Xavier, Bordes, Antoine, and Bengio, Yoshua. Deep sparse rectifier networks. In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume, volume 15, pp. 315–323, 2011.

Glorot，Xavier，Bordes，Antoine和Bengio，Yoshua。深度稀疏整流器网络。在第14届人工智能与统计国际会议论文集中。 JMLR W＆CP卷，第15卷，第315-323页，2011年。

Hafner, Roland and Riedmiller, Martin. Reinforcement learning in feedback control. Machine learning, 84(1-2):137–169, 2011.

哈夫纳，罗兰和里德米勒，马丁。反馈控制中的强化学习。机器学习，84（1-2）：137-169，2011。

Hasselt, Hado V. Double q-learning. In Advances in Neural Information Processing Systems, pp. 2613–2621, 2010.

Hasselt，Hado V.双重q-学习。 In Advances in Neural Information Processing Systems，pp.2613-2621,2010。

Heess, N., Hunt, J. J, Lillicrap, T. P, and Silver, D. Memory-based control with recurrent neural networks. NIPS Deep Reinforcement Learning Workshop (arXiv:1512.04455), 2015.

Heess，N.，Hunt，J. J，Lillicrap，T. P和Silver，D.基于记忆的控制与递归神经网络。NIPS深化强化学习研讨会（arXiv：1512.04455），2015。

Heess, Nicolas, Wayne, Gregory, Silver, David, Lillicrap, Tim, Erez, Tom, and Tassa, Yuval. Learning continuous control policies by stochastic value gradients. In Advances in Neural Information Processing Systems, pp. 2926–2934, 2015.

Heess，Nicolas，Wayne，Gregory，Silver，David，Lillicrap，Tim，Erez，Tom和Tassa，Yuval。通过随机值梯度学习连续控制策略。In Advances in Neural Information Processing Systems，pp.2926-2934,2015。

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

Ioffe，Sergey和Szegedy，Christian。批量标准化：通过减少内部协变量来加速深度网络培训。 arXiv预印本arXiv：1502.03167,2015。

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Kingma，Diederik和Ba，Jimmy。亚当：一种随机优化的方法。 arXiv预印本arXiv：1412.6980,2014。

Koutn´ık, Jan, Schmidhuber, J¨urgen, and Gomez, Faustino. Evolving deep unsupervised convolutional networks for vision-based reinforcement learning. In Proceedings of the 2014 conference on Genetic and evolutionary computation, pp. 541–548. ACM, 2014a.

Koutn'ık，Jan，Schmidhuber，Jüurgen和Gomez，Faustino。为基于视觉的强化学习开发深度无监督卷积网络。在2014年的遗传与进化计算会议论文集中，第541-548页。 ACM，2014a。

Koutn´ık, Jan, Schmidhuber, J¨urgen, and Gomez, Faustino. Online evolution of deep convolutional network for vision-based reinforcement learning. In From Animals to Animats 13, pp. 260–269. Springer, 2014b.

Koutn'ık，Jan，Schmidhuber，Jüurgen和Gomez，Faustino。用于视觉强化学习的深度卷积网络的在线演化。 In From Animals to Animats 13，pp.260-269。施普林格，2014b。

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pp. 1097–1105, 2012.

Krizhevsky，Alex，Sutskever，Ilya和Hinton，Geoffrey E.Imagenet分类与深卷积神经网络。在Advances in neural information processing systems，pp.1097-1105,2012中。

Levine, Sergey, Finn, Chelsea, Darrell, Trevor, and Abbeel, Pieter. End-to-end training of deep visuomotor policies. arXiv preprint arXiv:1504.00702, 2015.
莱文，谢尔盖，芬兰人，切尔西，达雷尔，特雷弗和阿比尔，彼得。深度视觉运动策略的端到端培训。 arXiv预印本arXiv：1504.00702,2015。

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.
MNIH，沃洛，Kavukcuoglu，科瑞，银，大卫·格雷夫斯，亚历克斯，Antonoglou，扬，奥德纳尔德，大安和Riedmiller，马丁。通过深入强化学习玩雅典娜。 arXiv预印本arXiv：1312.5602,2013。

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Humanlevel control through deep reinforcement learning. Nature, 518(7540):529–533, 2015.
Mnih，Volodymyr，Kavukcuoglu，Koray，Silver，David，Rusu，Andrei A，Veness，Joel，Bellemare，Marc G，Graves，Alex，Riedmiller，Martin，Fidjeland，Andreas K，Ostrovski，Georg等。通过深入强化学习来控制人的水平。 Nature，518（7540）：529-533,2015。

Prokhorov, Danil V, Wunsch, Donald C, et al. Adaptive critic designs. Neural Networks, IEEE Transactions on, 8(5):997–1007, 1997.
Prokhorov，Danil V，Wunsch，Donald C等人。适应性评论家设计Neural Networks，IEEE Transactions on，8（5）：997-1007,1997。

Schulman, John, Heess, Nicolas, Weber, Theophane, and Abbeel, Pieter. Gradient estimation using stochastic computation graphs. In Advances in Neural Information Processing Systems, pp. 3510– 3522, 2015a.
Schulman，John，Heess，Nicolas，Weber，Theophane和Abbeel，Pieter。使用随机计算图的梯度估计。In Advances in Neural Information Processing Systems，pp.3510-3522，2015a。

Schulman, John, Levine, Sergey, Moritz, Philipp, Jordan, Michael I, and Abbeel, Pieter. Trust region policy optimization. arXiv preprint arXiv:1502.05477, 2015b.
舒尔曼，约翰，莱文，谢尔盖，莫里茨，菲利普，约旦，迈克尔一世和彼得的阿比贝尔。信任域策略优化。 arXiv预印本arXiv：1502.05477，2015b。

Silver, David, Lever, Guy, Heess, Nicolas, Degris, Thomas, Wierstra, Daan, and Riedmiller, Martin. Deterministic policy gradient algorithms. In ICML, 2014.
银，大卫，莱弗，盖伊，赫斯，尼古拉斯，德格里斯，托马斯，维尔斯特拉，达安和里德米勒，马丁。确定性策略梯度算法。在ICML，2014年。

Tassa, Yuval, Erez, Tom, and Todorov, Emanuel. Synthesis and stabilization of complex behaviors through online trajectory optimization. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pp. 4906–4913. IEEE, 2012.
塔萨，尤瓦尔，埃雷兹，汤姆和托多罗夫，伊曼纽尔。通过在线轨迹优化合成和稳定复杂行为。在智能机器人和系统（IROS），2012年IEEE / RSJ国际会议上，第4906-4913页。 IEEE，2012。

Todorov, Emanuel and Li, Weiwei. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In American Control Conference, 2005. Proceedings of the 2005, pp. 300–306. IEEE, 2005.
托多罗夫，伊曼纽尔和李伟伟。约束非线性随机系统局部最优反馈控制的广义迭代lqg方法。在美国控制会议上，2005年会议录，第300-306页。 IEEE，2005。

Todorov, Emanuel, Erez, Tom, and Tassa, Yuval. Mujoco: A physics engine for model-based control. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pp. 5026– 5033. IEEE, 2012.
托多罗夫，伊曼纽尔，埃雷兹，汤姆和塔萨，尤瓦尔。 Mujoco：基于模型控制的物理引擎。在智能机器人和系统（IROS），2012年IEEE / RSJ国际会议上，第5026-5033页.IEEE，2012。

Uhlenbeck, George E and Ornstein, Leonard S. On the theory of the brownian motion. Physical review, 36(5):823, 1930.
Uhlenbeck，George E和Ornstein，Leonard S.关于布朗运动的理论。 Physical Review，36（5）：823,1930。

Wahlstr¨om, Niklas, Sch¨on, Thomas B, and Deisenroth, Marc Peter. From pixels to torques: Policy learning with deep dynamical models. arXiv preprint arXiv:1502.02251, 2015.
Wahlström，Niklas，Schöon，Thomas B和Deisenroth，Marc Peter。从像素到力矩：深度动态模型的政策学习。 arXiv预印本arXiv：1502.02251,2015。

Watkins, Christopher JCH and Dayan, Peter. Q-learning. Machine learning, 8(3-4):279–292, 1992.
沃特金斯，克里斯托弗JCH和达扬，彼得。 Q学习。机器学习，8（3-4）：279-292,1992。

Wawrzy´nski, Paweł. Real-time reinforcement learning by sequential actor–critics and experience replay. Neural Networks, 22(10):1484–1497, 2009.
Wawrzy'nski，Paweł。由连续的演员评论家和经验重播实时强化学习。神经网络，22（10）：1484-1497,2009。

Wawrzy´nski, Paweł. Control policy with autocorrelated noise in reinforcement learning for robotics. International Journal of Machine Learning and Computing, 5:91–95, 2015.
Wawrzy'nski，Paweł。自动相关噪声控制机器人强化学习策略。International Journal of Machine Learning and Computing，5：91-95，2015。

Wawrzy´nski, Paweł and Tanwani, Ajay Kumar. Autonomous reinforcement learning with experience replay. Neural Networks, 41:156–167, 2013.
Wawrzy'nski，Pawełand Tanwani，Ajay Kumar。凭借经验重播进行自主强化学习。神经网络，41：156-167,2013。

Supplementary Information: Continuous control with deep reinforcement learning
补充信息：通过深度强化学习进行持续控制

7 EXPERIMENT DETAILS
7实验细节

We used Adam (Kingma & Ba, 2014) for learning the neural network parameters with a learning rate of 10^{-4} and 10^{-3} for the actor and critic respectively. For Q we included L_2 weight decay of 10^{-2} and used a discount factor of $\gamma = 0.99$. For the soft target updates we used $\tau = 0.001$. The neural networks used the rectified non-linearity (Glorot et al., 2011) for all hidden layers. The final output layer of the actor was a tanh layer, to bound the actions. The low-dimensional networks had 2 hidden layers with 400 and 300 units respectively (≈ 130,000 parameters). Actions were not included until the 2nd hidden layer of Q. When learning from pixels we used 3 convolutional layers (no pooling) with 32 filters at each layer. This was followed by two fully connected layers with 200 units (≈ 430,000 parameters). The final layer weights and biases of both the actor and critic were initialized from a uniform distribution $[-3 \times 10^{-3}, 3 \times 10^{-3}]$ and $[3 \times 10^{-4}, 3 \times 10^{-4}]$ for the low dimensional and pixel cases respectively. This was to ensure the initial outputs for the policy and value estimates were near zero. The other layers were initialized from uniform distributions $[-\frac{1}{\sqrt{f}}, \frac{1}{\sqrt{f}}]$ where f is the fan-in of the layer. The actions were not included until the fully-connected layers. We trained with minibatch sizes of 64 for the low dimensional problems and 16 on pixels. We used a replay buffer size of 106.
我们使用Adam（Kingma＆Ba，2014）分别为演员和评论者学习神经网络参数，学习率为10^{-4}和10^{-3}。对于Q我们包括L_2重量衰减的10^{-2}，并使用$\gamma = 0.99$的折扣因子。对于软目标更新，我们使用$\tau = 0.001$。神经网络对所有隐藏层使用了整型非线性（Glorot et al。，2011）。演员最后的输出层是一个tanh层，用于限制动作。低维网络有2个隐藏层，分别有400个和300个单元（约130,000个参数）。在Q的第二个隐藏层之前不包括行动。当从像素学习时，我们使用3层卷积层（无池），每层有32层滤波器。随后是两个完全连接的层，有200个单元（约430,000个参数）。对于低维和像素情况，演员和评论者的最终层权重和偏差分别从均匀分布$[-3 \times 10^{-3}, 3 \times 10^{-3}]$和$[3 \times 10^{-4}, 3 \times 10^{-4}]$初始化。这是为了确保政策和价值估计的初始产出接近于零。其他层从均匀分布$[-\frac{1}{\sqrt{f}}, \frac{1}{\sqrt{f}}]$初始化，其中f是图层扇入。直到完全连接的层之后才会包含这些操作。我们用低维问题的64个小批次和16个像素进行了训练。我们使用了106的重播缓冲区大小。

For the exploration noise process we used temporally correlated noise in order to explore well in physical environments that have momentum. We used an Ornstein-Uhlenbeck process (Uhlenbeck & Ornstein, 1930) with $\theta = 0.15$ and $\sigma = 0.2$. The Ornstein-Uhlenbeck process models the velocity of a Brownian particle with friction, which results in temporally correlated values centered around 0.
对于探测噪声过程，我们使用时间相关噪声以便在具有动量的物理环境中进行良好的探测。我们使用了INNSLATEXT1009和$\sigma = 0.2$的Ornstein-Uhlenbeck工艺（Uhlenbeck＆Ornstein，1930）。Ornstein-Uhlenbeck过程利用摩擦模拟布朗粒子的速度，这导致时间相关值集中在0附近。

8 PLANNING ALGORITHM
8规划算法

Our planner is implemented as a model-predictive controller (Tassa et al., 2012): at every time step we run a single iteration of trajectory optimization (using iLQG, (Todorov & Li, 2005)), starting from the true state of the system. Every single trajectory optimization is planned over a horizon between 250ms and 600ms, and this planning horizon recedes as the simulation of the world unfolds, as is the case in model-predictive control.
我们的计划是作为模型预测控制器实现的（Tassa等，2012）：在每一步我们运行一次迭代的轨迹优化（使用iLQG，（Todorov＆Li，2005）），从真实状态开始系统。每个单一的轨迹优化计划在250ms和600ms之间的一个范围内，随着模拟世界的展开，这个计划范围将减少，就像模型预测控制的情况一样。

The iLQG iteration begins with an initial rollout of the previous policy, which determines the nominal trajectory. We use repeated samples of simulated dynamics to approximate a linear expansion of the dynamics around every step of the trajectory, as well as a quadratic expansion of the cost function. We use this sequence of locally-linear-quadratic models to integrate the value function backwards in time along the nominal trajectory. This back-pass results in a putative modification to the action sequence that will decrease the total cost. We perform a derivative-free line-search over this direction in the space of

action sequences by integrating the dynamics forward (the forwardpass), and choose the best trajectory. We store this action sequence in order to warm-start the next iLQG iteration, and execute the first action in the simulator. This results in a new state, which is used as the initial state in the next iteration of trajectory optimization.
iLQG迭代开始于先前策略的首次推出，该策略确定标称轨迹。我们使用仿真动态的重复样本近似轨迹每一步周围动力学的线性展开，以及成本函数的二次展开。我们使用这个序列的局部线性二次模型来将价值函数沿着标称轨迹向后整合。这种回传导致对动作序列进行假定修改，从而降低总成本。我们通过整合动态前进（前进通道）并选择最佳轨迹，在动作序列的空间中执行这个方向上的无导数线搜索。我们存储这个动作序列，以便热启动下一个iLQG迭代，并在模拟器中执行第一个动作。这导致了一个新的状态，它被用作下一次轨迹优化迭代的初始状态。

9 ENVIRONMENT DETAILS
9环境细节

9.1 TORCS ENVIRONMENT
9.1 TORCS环境

For the Torcs environment we used a reward function which provides a positive reward at each step for the velocity of the car projected along the track direction and a penalty of -1 for collisions. Episodes were terminated if progress was not made along the track after 500 frames.
对于Torcs环境，我们使用了奖励函数，该函数在每个步骤对沿着轨道方向投射的汽车的速度提供正面奖励，并对碰撞的-1惩罚。如果在500帧之后没有沿着轨道进行，则情节终止。

9.2 MUJOCO ENVIRONMENTS
9.2 MUJOCO环境

For physical control tasks we used reward functions which provide feedback at every step. In all tasks, the reward contained a small action cost. For all tasks that have a static goal state (e.g. pendulum swingup and reaching) we provide a smoothly varying reward based on distance to a goal state, and in some cases an additional positive reward when within a small radius of the target state. For grasping and manipulation tasks we used a reward with a term which encourages movement towards the payload and a second component which encourages moving the payload to the target. In locomotion tasks we reward forward action and penalize hard impacts to encourage smooth rather than hopping gaits (Schulman et al., 2015b). In addition, we used a negative reward and early termination for falls which were determined by simple threshholds on the height and torso angle (in the case of walker2d).
对于物理控制任务，我们使用奖励功能，在每一步都提供反馈。在所有任务中，奖励只包含一小部分行动成本。对于具有静态目标状态的所有任务（例如钟摆摆动和到达），我们提供基于到目标状态的距离的平稳变化的奖励，并且在某些情况下，在目标状态的小半径范围内提供额外的积极奖励。对于抓握和操纵任务，我们使用奖励来鼓励向有效载荷移动，第二个组件鼓励将有效载荷移动到目标。在运动任务中，我们奖励前进行动并惩罚艰难的影响，以鼓励平稳而不是跳跃的步伐（Schulman等，2015b）。此外，我们使用负面报酬和提前终止跌倒，这是通过身高和躯干角度的简单阈值确定的（以walker2d为例）。

Table 2 states the dimensionality of the problems and below is a summary of all the physics environments.
表2列出了问题的维度，下面是所有物理环境的总结。

task name	$\dim(\mathbf{s})$	$\dim(\mathbf{a})$	$\dim(\mathbf{o})$
blockworld1	18	5	43
blockworld3da	31	9	102
canada	22	7	62
canada2d	14	3	29
cart	2	1	3
cartpole	4	1	14
cartpoleBalance	4	1	14
cartpoleParallelDouble	6	1	16
cartpoleParallelTriple	8	1	23
cartpoleSerialDouble	6	1	14
cartpoleSerialTriple	8	1	23
cheetah	18	6	17
fixedReacher	10	3	23
fixedReacherDouble	8	2	18
fixedReacherSingle	6	1	13
gripper	18	5	43
gripperRandom	18	5	43
hardCheetah	18	6	17
hardCheetahNice	18	6	17

hopper	14	4	14
hyq	37	12	37
hyqKick	37	12	37
movingGripper	22	7	49
movingGripperRandom	22	7	49
pendulum	2	1	3
reacher	10	3	23
reacher3daFixedTarget	20	7	61
reacher3daRandomTarget	20	7	61
reacherDouble	6	1	13
reacherObstacle	18	5	38
reacherSingle	6	1	13
walker2d	18	6	41

Table 2: Dimensionality of the MuJoCo tasks: the dimensionality of the underlying physics model $\dim(s)$ *, number of action dimensions* $\dim(a)$ *and observation dimensions* $\dim(o)$ *.*

表2：MuJoCo任务的维度：底层物理模型 $\dim(s)$ *的维度，动作维度数量* $\dim(a)$ *和观察维度* $\dim(o)$ *。*

task name	**Brief Description**
blockworld1	Agent is required to use an arm with gripper constrained to the 2D plane to grab a falling block and lift it against gravity to a fixed target position.

blockworld3da Agent is required to use a human-like arm with 7-DOF and a simple gripper to grab a block and lift it against gravity to a fixed target position.
blockworld3da Agent需要使用一个带有7自由度的人形手臂和一个简单的抓手来抓住一个块，并将其抵抗重力提升到一个固定的目标位置。

canada Agent is required to use a 7-DOF arm with hockey-stick like appendage to hit a ball to a target.
加拿大代理商需要使用带有曲棍球棒附件的7自由度手臂将球击中目标。

canada2d Agent is required to use an arm with hockey-stick like appendage to hit a ball initialzed to a random start location to a random target location.
canada2d Agent需要使用像曲棍球棒一样的手臂将一个球初始化为一个随机起始位置，并将其移动到一个随机目标位置。

cart Agent must move a simple mass to rest at 0. The mass begins each trial in random positions and with random velocities.
推动者必须将一个简单的质量移动到0处。质量以随机位置和随机速度开始每次试验。

cartpole The classic cart-pole swing-up task. Agent must balance a pole attached to a cart by applying forces to the cart alone. The pole starts each episode hanging upside-down.
手推车经典的手推车摆动任务。代理商必须通过将力量单独施加到购物车上来平衡连接到购物车的杆子。杆开始每一集倒挂。

cartpoleBalance The classic cart-pole balance task. Agent must balance a pole attached to a cart by applying forces to the cart alone. The pole starts in the upright positions at the beginning of each episode.
cartpoleBalance经典的手推车平衡任务。代理商必须通过将力量单独施加到购物车上来平衡连接到购物车的杆子。杆在每集开始时以直立位置开始。

cartpoleParallelDouble Variant on the classic cart-pole. Two poles, both attached to the cart, should be kept upright as much as possible.
cartpoleParallelDouble经典手推车杆上的变体。两个连接在车上的杆应尽可能地保持直立。

cartpoleSerialDouble Variant on the classic cart-pole. Two poles, one attached to the cart and the second attached to the end of the first, should be kept upright as much as possible.
cartpoleSerialDouble经典手推车杆上的变体。两根杆，一根连接在车上，另一根连接到第一根杆上，应尽可能保持直立。

cartpoleSerialTriple Variant on the classic cart-pole. Three poles, one attached to the cart, the second attached to the end of the first, and the third attached to the end of the second, should be kept upright as much as possible.

cartpoleSerialTriple变体在经典手推车杆。三个杆，一个连接到车上，第二个连接到第一个末端，第三个连接到第二个末尾，应尽可能保持直立。

cheetah The agent should move forward as quickly as possible with a cheetahlike body that is constrained to the plane. This environment is based very closely on the one introduced by Wawrzy´nski (2009); Wawrzy´nski & Tanwani (2013).
猎豹代理商应该尽快向前移动，并与被限制在飞机上的猎豹般的身体接触。这个环境非常接近Wawrzy'nski（2009）提出的环境; Wawrzy'nski＆Tanwani（2013）。

fixedReacher Agent is required to move a 3-DOF arm to a fixed target position.
需要固定的固定剂才能将三自由度臂移动到固定的目标位置。

fixedReacherDouble Agent is required to move a 2-DOF arm to a fixed target position.
必须使用xedReacherDouble Agent将2-DOF臂移至固定的目标位置。

fixedReacherSingle Agent is required to move a simple 1-DOF arm to a fixed target position.
fi xedReacherSingle Agent需要移动一个简单的1-DOF手臂到固定的目标位置。

gripper Agent must use an arm with gripper appendage to grasp an object and manuver the object to a fixed target.
抓手代理必须使用带抓手附肢的手臂来抓取物体并将物体固定在固定的目标上。

gripperRandom The same task as gripper except that the arm object and target position are initialized in random locations.
gripperRandom与抓手相同的任务，只是手臂对象和目标位置在随机位置初始化。

hardCheetah The agent should move forward as quickly as possible with a cheetahlike body that is constrained to the plane. This environment is based very closely on the one introduced by Wawrzy´nski (2009); Wawrzy´nski & Tanwani (2013), but has been made much more difficult by removing the stabalizing joint stiffness from the model.
hardCheetah代理商应尽可能快地向前移动，并与被限制在飞机上的猎豹般的身体接触。这个环境非常接近Wawrzy'nski（2009）提出的环境; Wawrzy'nski＆Tanwani（2013），但是通过从模型中消除了稳定的关节刚度使得困难得多。

hopper Agent must balance a multiple degree of freedom monoped to keep it from falling.
料斗代理必须平衡单独的多重自由度以防止其下落。

hyq Agent is required to keep a quadroped model based on the hyq robot from falling.
hyq Agent需要保持基于Hyq机器人的四足模型不会掉落。

movingGripper	Agent must use an arm with gripper attached to a moveable platform to grasp an object and move it to a fixed target.
movingGripperRandom	The same as the movingGripper environment except that the object position, target position, and arm state are initialized randomly.
pendulum	The classic pendulum swing-up problem. The pendulum should be brought to the upright position and balanced. Torque limits prevent the agent from swinging the pendulum up directly.
reacher3daFixedTarget	Agent is required to move a 7-DOF human-like arm to a fixed target position.
reacher3daRandomTarget	Agent is required to move a 7-DOF human-like arm from random starting locations to random target positions.
reacher	Agent is required to move a 3-DOF arm from random starting locations to random target positions.
reacherSingle	Agent is required to move a simple 1-DOF arm from random starting locations to random target positions.
reacherObstacle	Agent is required to move a 5-DOF arm around an obstacle to a randomized target position.

walker2d	Agent should move forward as quickly as possible with a bipedal walker constrained to the plane without falling down or pitching the torso too far forward or backward.