南京大学

本 科 毕 业 论 文

院　　系 工程管理学院

专　　业 自动化

题　　目 深度强化学习中基于优先级的经验回放机制研究

年　　级 2013　学 号 131270027

学生姓名 侯跃南

指导教师 陈春林　职 称 教授

提交日期

南京大学本科生毕业论文（设计、作品）中文摘要

题目：深度强化学习中基于优先级的经验回放机制研究

院系：工程管理学院

专业：自动化

本科生姓名：侯跃南

指导教师（姓名、职称）：陈春林 教授

摘要：经验回放机制使得强化学习算法能够利用过去的经验同时防止输入的数据信息有较强的时序相关性。最近，名为 DDPG（深度确定性策略梯度算法）的深度强化学习算法解决了许多动力学的仿真任务，而经验回放机制就是其中一个重要的组成部分。不过，经验回放机制不能够充分使用获得的经验因为它认为所有的经验的价值相同并采取了均匀采样的策略。为了提高经验的使用效率，我们提出在 DDPG 算法中将均匀经验回放改为基于优先级的经验回放。我们将改进的 DDPG 算法和原 DDPG 算法在 OpenAI Gym 仿真平台上进行了比较。我们测试了多个仿真任务，并选取算法获得的奖励值、稳定性和对多个超参的敏感程度作为评价指标。实验结果表明，改进的 DDPG 算法明显优于原算法。并且，在这些仿真任务上，我们的算法的表现也好于目前最主流的算法如 Q-prop、TRPO 等。这些结果有力地证明我们提出的算法的有效性。

关键词：经验回放，深度强化学习，深度确定性策略梯度算法，基于优先级的经验回放

南京大学本科生毕业论文（设计、作品）英文摘要

THESIS: The Study of Combining Deep Reinforcement Learning with Prioritized Experience Replay

DEPARTMENT: School of Management and Engineering

SPECIALIZATION: Automation

UNDERGRADUATE: Yuenan Hou

MENTOR: Chunlin Chen

ABSTRACT: Experience replay mechanism helps reinforcement learning algorithms make use of the previous experiences as well as prevent input data from being highly correlated. Recently, a deep reinforcement learning algorithm called deep deterministic policy gradient (DDPG) has solved many simulated physics tasks with experience replay mechanism being a significant component. However, this approach can hardly make full use of the experience agents obtain from the environment for it considers all experience to be of same value and samples experience uniformly. In order to improve the efficiency of experience usage, we propose to replace uniform experience replay with prioritized experience replay. Our algorithm is compared with original DDPG algorithm in five tasks in the OpenAI Gym, a testbed for reinforcement learning algorithms. In the experiment, DDPG with prioritized experience replay significantly outperforms original DDPG in terms of rewards, training time, training stability, sensitiveness to some hyperparameters (i.e., the

size of replay buffer, minibatch and the updating rate of the target network). Our algorithm can achieve better performance in three tasks compared with some state-of-the-art algorithms like Q-prop and TRPO, which directly proves the effectiveness of our proposed scheme.

KEY WORDS: Experience Replay, Deep Reinforcement Learning, Deep Deterministic Policy Gradient, Prioritized Experience Replay

目录

1 绪论 1
1.1 研究背景 1
1.1.1 深度学习 1
1.1.2 强化学习 2
1.1.3 深度强化学习 3
1.1.4 经验回放机制 3
1.2 章节安排 4
2 深度强化学习 5
2.1 深度学习 5
2.2 强化学习 7
2.3 深度强化学习 10
2.3.1 离散动作领域-深度 Q 网络 10
2.3.2 连续动作领域-深度确定性策略梯度网络 12
3 基于优先级的经验回放机制 15
3.1 传统的经验回放机制 15
3.2 基于优先级的经验回放机制 16
3.2.1 经验优先级的确定 16
3.2.2 随机优先级采样 16
3.2.3 重要性采样系数的加入 18
3.2.4 数学原理解释 19
3.2.5 完整的算法 21
4 实验 23
4.1 实验的配置 24
4.2 倒立摆仿真实验 25
4.3 二级倒立摆仿真实验 29
4.4 猎豹式奔跑仿真实验 30
4.4.1 最大步数改变对实验的影响 31
4.5 单腿跳仿真实验 32
4.6 跑步仿真实验 32
4.7 超参变化对实验的影响 34
4.8 和其它算法的比较 36
5 结论与展望 37
6 相关的科研成果 38

1 绪论

作为人工智能领域的主流算法，深度强化学习在 Atari 游戏[1-2]、连续控制[3-4]、手写字体识别[5]等许多领域得到广泛应用。不过，作为深度强化学习算法的一个核心部分，经验回放机制[6-7]认为所有的经验的价值相同并选择随机均匀采样。这样，那些更加有价值的经验和一般的经验的回放频次相同，从而降低了经验的使用效率。我们提出用基于优先级的经验回放机制[8-9]来替换传统的经验回放机制，核心就是更加高频地回放那些更加有价值的经验，而评价经验价值的指标则是经验的 TD-error（Temporal Difference error，时间差分偏差）。我们测试的基本算法是深度确定性策略梯度算法[3]（DDPG），测试领域是连续控制领域。

1.1 研究背景

1.1.1 深度学习

一个神经网络（neural network）包含很多层（layer），而每一层又由很多神经元（neuron）构成。深度学习的本质是通过这种深层堆叠的网络结构来从输入数据中自动学习其特征表示，即达到所谓的数据降维的作用。

神经网络的起源可以追溯到上世纪 40 年代。那个时候，著名心理学家 Warren Mcculloch 在其论文[10]中首次提出人工神经网络的概念及其数学模型，这个时候神经网络或者说是深度学习的雏形就已经慢慢显现。到 50 年代中期，最简单的神经网络——感知机[11]（perceptron）的概念被提出。在单层感知机的输入输出间加入隐含层(hidden layer)，就得到了多层感知机(multi-layer perceptron)。不过，由于多层感知机的隐含层的权重没有合适的方法来训练调整，所以在当时并不被看好。神经网络的真正崛起是在 80 年代，因为此时反向传播算法[12]（back propagation，BP）被提出用来调整隐含层的权重并取得了不错的成果，此时深度学习开始萌芽。但是，BP 算法在反向传播的过程中利用梯度下降算法对网络权重进行调整时容易出现梯度消散（gradient diffusion）、易于陷入局部最优（local optimal）等问题，并且当网络层数加深，这些问题会更加严重。这些问题直到 2006 年 Geoffrey Hinton 提出逐层训练、整体微调的网络模型训练方

法才得到解决[13]，此时深度学习才进入高速发展的时期。随后，深度信念网络[14]（Deep Belief Network，DBN）、递归神经网络[15]（Recurrent Neural Network，RNN）、卷积神经网络[16]（Convolutional Neural Network，CNN）也被相继提出并在手写字体识别[5]、图片分类[17]、自然语言处理[18]、目标检测[19]等领域取得了相当显著的成功。最近，深度学习和强化学习相结合，即利用深度网络作为强化学习的Q函数、策略函数的函数逼近器，使得强化学习算法能够推广到高维状态和动作空间[1-4]。

1.1.2 强化学习

强化学习就是一个智能体学习一个从状态到动作的映射，即策略函数，使其获得的长期奖励最大化[20]。强化学习的本质是试错学习（trial and error）和延迟回报（delayed reward）。试错学习即意味着智能体完成一件事需要尝试很多次，然后从获得的很多经验中学习到正确的策略。但是这种学习是延迟的，即我们不会时时刻刻纠正智能体的行为，相反而是根据它的表现给一个奖励值，指引它来完成我们期望它完成的事情。在强化学习中，由于我们需要评价当前策略是否最优，所以动态规划法（Dynamic Programming，DP）、蒙特卡罗方法（Monte Carlo，MC）和时间差分法（Temporal Difference，TD）被相继提出。强化学习过程通常被建模为马尔科夫决策过程（Markov Decision Process，MDP）和半马尔科夫决策过程(Semi-Markov Decision Process,SMDP)。Q学习[21](Q-learning)通常被用于评估当前采取的动作的价值，而贝尔曼公式（Bellman Equation）则被用于更新学到的Q值。作为最经典的强化学习算法，TD算法最早由Sutton提出，Q学习最早是由Chris Watkins在其博士论文“Learning from Delayed Rewards”[22]中首次提出。TD算法是在线更新算法，即是拿同一个策略来更新迭代，而Q学习是离线更新算法，即选择下个状态下Q值最大的动作来迭代更新。而在线的TD算法就是SARSA。最近，Actor-Critic算法[23]被提出，用于分离评价动作价值的网络和策略网络。在连续控制领域，确定性策略梯度算法和深度神经网络结合，得到了深度确定性策略梯度算法。策略梯度算法[24]（policy gradient）也被很好的利用起来，用于更新策略网络。

1.1.3 深度强化学习

过去，人们曾试图将强化学习算法和神经网络相结合，不过却都以失败而告终。由于深度神经网络为非线性模型，在训练过程中网络参数变化幅度大，容易振荡甚至发散。不过，最大的问题在于强化学习得到的输入数据间存在很强的时序相关性，而这种时序相关性又和那些更新神经网络的梯度算法的前提相冲突[1]。这时，目标网络训练法[1]（target network）和经验回放机制[6-7]（experience replay mechanism）这两种方法被提出用来克服上面提到的问题。目标网络法就是每隔一定步数将待训练的网络做一个拷贝，然后待训练的网络以这个拷贝网络作为目标，不断更新自己的网络参数。经验回放机制将在下面介绍。同时，为了提高网络训练的稳定性，批正则化[25]（batch normalization）被提出用于预处理输入每层网络的数据，以达到限制每层网络权重的变化范围的目的。这样，结合了上述几个方法，强化学习和深度学习很好地结合，形成了深度强化学习算法（Deep Reinforcement Learning，DRL）。

1.1.4 经验回放机制

经验回放机制就是拿一块有限大小的内存来存储智能体之前和环境交互得到的经验（经验就是(s_t, a_t, r_t, s_{t+1})元组），这块内存也叫经验池（replay buffer）。然后，每次神经网络需要输入时，从这个经验池中随机选择一个 minibatch 数目的经验来更新网络。这种做法有两个好处，一个是由于之前获得的经验也机会被再次学到，所以经验的使用效率得到提高。二是随机选择经验回放的做法使得输入经验间的时序相关性被打破。不过，由于是随机选择经验回放，那些更加有价值的经验和一般的经验便被同样的频率被回放，所以经验的学习效率便不会太高。而本文正是考虑到传统的经验回放机制的这个不足，提出使用基于优先级的经验回放机制。

1.2 章节安排

本文在第二章介绍深度学习、强化学习以及深度强化学习的基本知识，同时简要介绍了两种经典的深度强化学习算法。在第三章我们将简要介绍传统的经验回放机制，同时具体介绍基于优先级的经验回放机制，并给出改进算法的数学依据。第四章我们将介绍我们做的对比试验，即传统的经验回放机制和基于优先级的经验回放机制，测试的基础算法是深度确定性策略梯度算法。第五章和第六章则分别给出了结论展望和相关的科研成果。

2 深度强化学习

2.1 深度学习

深度学习和强化学习只是机器学习领域的一个部分，因此在介绍深度学习和强化学习之前，我们有必要提一下机器学习[26]（machine learning)。机器学习就是一个通过直接使用数据的方式来设计和最优化信息处理系统的方法。在现实世界中，由于训练数据有限并且不可能包含所有可能的场景，这个信息处理系统需要学着去在训练数据没有包含的那部分场景里有较好的表现。在这种情况下，过拟合（over-fitting）是机器学习领域中的一个非常普遍和棘手的问题，它的一般表现就是所训练的系统的表现可以严格符合训练集的要求，但却不能很好地在那些没有见过的输入数据中有较好的表现，即泛化性能(generalization)差。

机器学习通常分为监督学习（supervised learning）、强化学习（reinforcement learning）和非监督学习（unsupervised learning)。监督学习系统就是学习从一个有期望输出的数据集中学得一个输入-输出的映射，也就是它们通过期望输出可以明确知道该怎么做（即训练数据有标签)。强化学习系统则相反，它是通过和环境的交互并获得一个反馈（feedback）来学着该怎么做以达到期望的目标，这个反馈通常是以一个数值信号给出。非监督学习则是关于在数据中寻找数据之间存在的类别关系的，通常做聚类分析（clustering analysis)。其中，深度学习和其所使用的神经网络是目前最主流的监督学习方法。

深度学习通常使用神经网络（neural network）来学习从输入到输出的一个复杂的映射关系（mapping)，即利用神经网络作为函数逼近器（function approximator）来拟合一个复杂的函数关系。神经网络通常由许多神经元构成，这些神经元构成神经网络的每一层（layer)，如下图所示：

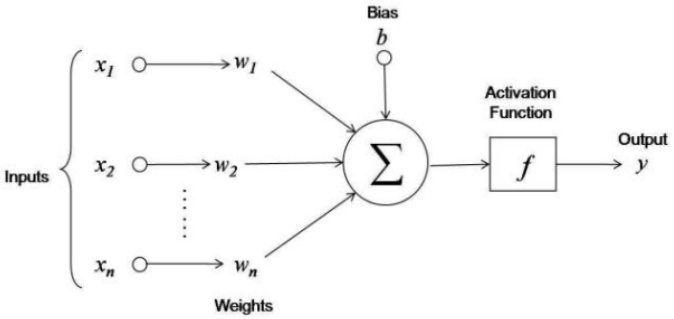


图 2-1：神经元示意图[48]

其中，输入量 x（由 x_1、x_2、……x_n组成）乘以权重 w（weight）加上偏差量 b（bias），通过一个激活函数 f（activation function）得到了每个神经元的输出 y：

$$y = f(\sum_i w_i x_i + b) \qquad (1)$$

许多神经元并行排列就形成了神经网络的一层。下图展示了包含一个输入层、一个隐层和一个输出层的神经网络。

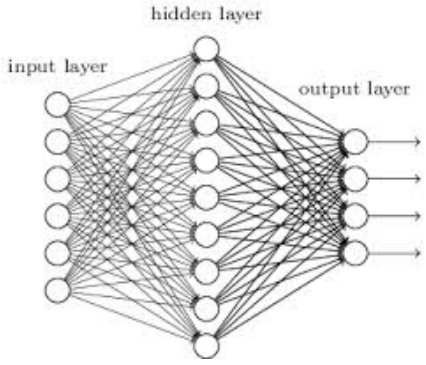
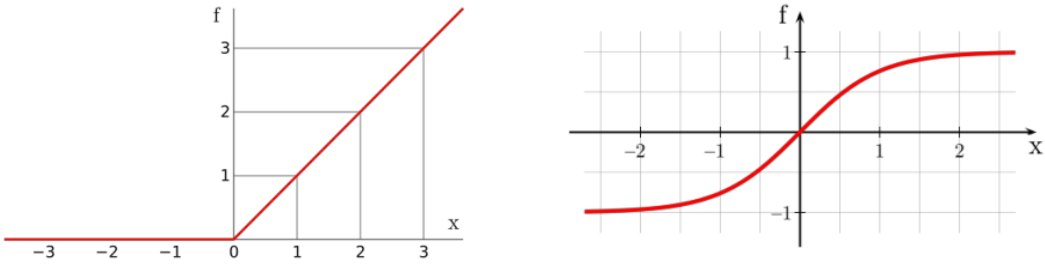


图 2-2：神经网络示意图

这里，激活函数通常是线性整流单元（rectified linear unit, ReLU）、双曲正切函数（tanh）等（见下图）。可以看出，激活函数在整个函数作用域上是单调增加的。



(a) Rectified Linear Unit (ReLU): $f(x) = \max(0, x)$. **(b)** Hyperbolic Tangent (tanh): $f(x) = \tanh(x)$.

图 2-3：激活函数示意图[48]

通常神经网络是随机初始化的，然后通过一个代价函数（cost function）来在一个数据集上训练的。代价函数通常是描述期望输出（目标输出 t_i）和网络实际输出（y_i）之间的距离的。一个常见简单的代价函数就是均方误差（mean squared error）：

$$f_{cost} = \frac{1}{n}\sum_i (y_i - t_i)^2 \qquad (2)$$

其中 n 为训练数据的个数。

训练一个神经网络的本质在于最优化网络参数 $\theta = (w_1, b_1, ..., w_n, b_n)$ 来最小化训练数据集上的代价函数。然而，训练一个神经网络是十分具有挑战性的。首先，由于参数的空间是非凸的，所以利用一般的梯度下降算法会导致网络参数的解收敛到局部最优。其次，参数的个数和网络的复杂程度成正比。由于深层的、复杂的网络通常需要去学习并拟合一个复杂的函数，从计算时间和数据的角度来看，训练过程的要求会非常高。不过，事实上如果使用随机梯度下降算法，那么网络的训练将会变得容易和方便。

2.2 强化学习

强化学习就是一个智能体（agent）以一种合适的方式和环境交互来最大化长期奖励值的过程。在强化学习的场景下，agent 不是被教着去完成任务，相反是要通过反馈的奖励信号来学着去做一个任务。状态、动作、奖励信号是一个强化学习的基本组成。状态用来刻画环境和 agent 组成的整体，动作表明 agent 对环境施加的作用，此时环境会给 agent 一个奖励信号和一个新的状态。如图所

示，对于当前状态 s，agent 执行一个动作 a，获得一个奖励 r，并得到一个新的状态 s'。强化学习的核心是学习从状态 s 到动作 a 的映射，即当前状态下应当选择什么动作才可以最大化奖励，亦即策略（policy）。这里，策略可以是随机的，即一个动作对状态的概率分布（probability distribution），也可以是确定性的（deterministic）。通常，强化学习的目标是学习到一个合适的策略使得未来获得的累积奖励总和最大：

$$R_t = r_t + r_{t+1} + r_{t+2} + \ldots\ldots + r_T = \sum_{i=t}^{T} r_i \qquad (3)$$

T 是一个强化学习过程结束的有限时刻。实际上，我们一般使用带有折扣的未来奖励和，即：

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots\ldots + \gamma^{T-t} r_T = \sum_{i=t}^{T} \gamma^{i-t} r_i \qquad (4)$$

其中，折扣因子 $\gamma \in (0,1)$ 保证当时间跨度为无限（infinite）时奖励值能够收敛。

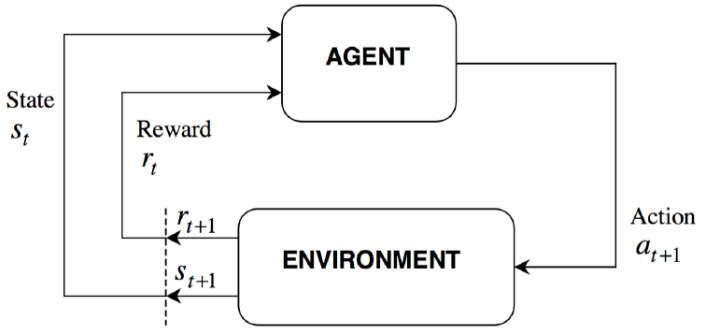


图 2-4：强化学习示意图[20]

在强化学习中，通常我们会用一个价值函数 $V^{\pi}(s)$ 来描述在状态 s 下一直执行策略 π 可以获得的未来奖励总和的期望，即：

$$V^{\pi}(s_0) = E[\sum_{i=0}^{T} \gamma^i r_{i+1} \mid s = s_0] \qquad (5)$$

该式表明从状态 s_0 出发所能够获得的折扣的未来奖励的期望，又被称为 γ 折扣累积奖赏。这种方法也叫做基于值（value-based）的算法。同样，我们定义一个动作-值函数（action-value function）$Q^{\pi}(s,a)$ 来描述在状态 s 下采取动作 a 后一直执行策略 π 可以获得的未来奖励的总和的期望，即：

$$Q^{\pi}(s_0,a_0)=E[\sum_{i=0}^{T}\gamma^i r_{i+1}\,|\,s=s_0,a=a_0] \tag{6}$$

而优势函数（advantage function）$A^{\pi}(s,a)$则连接了价值函数和动作-值函数：

$$A^{\pi}(s_0,a_0)=Q^{\pi}(s_0,a_0)-V^{\pi}(s_0) \tag{7}$$

对于确定性策略，动作-值函数可以通过贝尔曼公式（Bellman Equation）进行迭代求解：

$$Q^{\pi}(s_t,a_t)=E[r+\gamma Q^{\pi}(s_{t+1},a_{t+1})]|_{a_{t+1}=\pi(s_{t+1})} \tag{8}$$

知道对于每个状态-动作对的最优动作-值函数$Q^{\pi^*}(s_t,a_t)$以后，我们所要找的最优策略π^*就是在一个特定的状态下，每次选择 Q 值最高的那个动作，也就是：

$$\pi^*(s_t)=\arg\max_a Q^{\pi^*}(s_t,a_t) \tag{9}$$

事实上，刚开始我们并不需要知道$Q^{\pi^*}(s_t,a_t)$或者π^*。然而，即使从一个随机的动作-值函数 Q 开始训练，也可以通过迭代更新的方式（即利用贝尔曼公式）来得到最优$Q^{\pi^*}(s_t,a_t)$和最优策略π^*。这种方法是深度 Q 网络的一个核心元素，在后面的章节里我们会加以详述。

在连续动作空间中，因为有无穷多个动作需要去选择，所以此时对 Q 函数进行优化是不可行的。即使对相对低维度的动作空间，将动作空间进行简单的离散化也是很棘手的。例如对于一个 7 自由度的系统（例如人手），假设对每个关节只有伸直、弯曲和握紧三个动作，即使这样粗糙的建模，其需要搜索的动作空间也包含 3^7=2187 个动作。如果再细化一下的话，那么动作数目会指数型增加（维数灾难问题）。为了克服这个困难，也许直接去学习一个参数化的策略会比学习一个值函数更加有效，也就是所谓的基于策略（policy-based）的算法。一个很典型的方法就是策略梯度算法（policy gradient）。策略梯度算法利用期望回报 R_t 对策略参数 w 的导数$\nabla_w R_t$作为梯度，直接优化策略参数。而期望回报对策略参数的导数$\nabla_w R_t$则可以从样本中直接估计得出。

将基于值的算法和基于策略的算法相结合就得到了动作-值（actor-critic）

算法。这些方法利用了一个参数化的策略（actor）和一个用于改善策略的值函数估计器（critic）。一个比较典型的用于学习确定性策略的动作-值算法是确定性策略梯度算法（deterministic policy gradient）。确定性策略梯度算法使用一个可求导的动作-值函数逼近器，并通过使用输出的 Q 值关于动作 a 的导数 dQ/da 来得到策略梯度。此时策略梯度为$\nabla_w Q = \nabla_a Q \cdot \nabla_w \pi$。深度确定性策略梯度算法则是将确定性策略梯度算法和神经网络相结合，将在后面的章节详述。

2.3 深度强化学习

正如前面章节提到的那样，强化学习十分依赖拟合 Q 函数（基于值的算法）或是策略参数化（基于策略的算法）的过程。在上面两种情况中，函数逼近器的广泛使用使得强化学习能够用在更加复杂的问题中。神经网络在监督学习中已经被很成功地用作函数逼近器并且它们是可导的，这种性质对于很多强化学习算法来说都十分有用。在这个章节，我们集中讲述两种深度强化学习方法：一种是深度 Q 网络（Deep Q-Network, DQN），一个基于值的算法，能够从图像信息中玩 49 种 Atari 游戏并且超过人类玩家的水平[1-2]。第二个就是深度确定性策略梯度算法[3-4]（Deep Deterministic Policy Gradient），是一个动作-值算法，也是 DQN 在连续动作领域的一个变体。

2.3.1 离散动作领域-深度 Q 网络

深度 Q 网络的输入是像素信息，输出是控制游戏杆的动作，如上、下、左、右、开火等，即是在离散的动作域上的。从本质上来看，深度 Q 网络是一个基于值的算法，它用一个神经网络 Q(s,a|θ)来拟合最优的动作-值函数 Q^*(s,a)。Q 函数的训练目标是 $y_i = r_i + \gamma Q(s_{i+1}, \pi(s_{i+1}|\theta')|\theta')$，网络权重的调整是通过最小化其损失函数 L(θ)，也即：

$$\begin{aligned} & L(\theta) \\ & = (r_i + \gamma Q(s_{i+1}, \pi(s_{i+1}|\theta')|\theta') - Q(s_i, a_i|\theta))^2 \\ & = (r_i + \gamma \max_a Q(s_{i+1}, a|\theta') - Q(s_i, a_i|\theta))^2 \end{aligned} \tag{10}$$

然而，由于 Q 函数的训练目标依赖其自身，所以这样会导致训练过程的不稳定甚至发散。这里我们采用了目标网络法（target network），即将原来的 Q 函数做一个拷贝，得到目标 Q 函数。然后，原来的 Q 函数以这个目标 Q 函数为目标来训练并不断更新，目标 Q 函数则是每隔一定的步长 C 从原来的 Q 函数拷贝得到。这样，训练过程存在的耦合性就会被解除，从而大大提高训练过程的稳定性。

另外一个不稳定性因素在于输入网络的数据之间存在强烈的时序相关性。因为这些数据，也即(s_i, a_i, r_i, s_{i+1})，它们是从同一个轨迹（trajectory）中得到的。再者，当 Q 函数改变时，策略和数据分布也会明显改变。深度 Q 网络通过经验回放机制（experience replay mechanism）很好地解决了这个问题，也即从一个经验池中随机选择一个最小批（minibatch）的数据(s_i, a_i, r_i, s_{i+1})来训练网络，这样输入数据间的时序相关性被打破，从而平滑了输入分布的变化。这种做法同时增加了数据的使用效率，因为每个经验会被使用多次。关于经验回放机制，我们将在后面的章节详述。

最后，为了确保充足的搜索，一个简单的 ε 贪心（ε-greedy）策略，也就是 $\pi(s|\theta)=\arg\max_a Q(s,a|\theta)$。这种异策略（off-policy）方法是可行的因为算法只需要在片段（transition）上学习而不需要在整个运动轨迹（trajectory）上学习。完整的算法见下图。

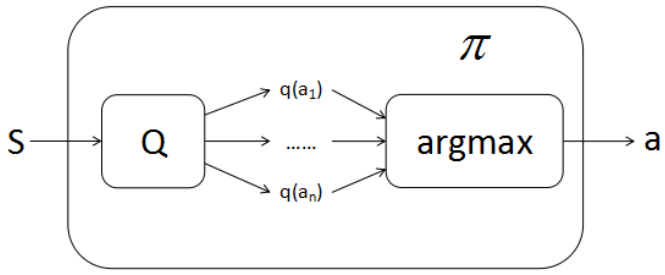


图 2-5：深度 Q 网络结构示意图

表 1：深度 Q 网络算法框架

深度 Q 网络算法

初始化经验池 D

初始化 Q 网络，其权重随机初始化为 θ

初始化目标 Q 网络，其权重 θ’ = θ

得到初始状态 s_1

for t=1 to T do

根据 ε 贪心策略选择动作 a_t

执行动作 a_t，得到奖励 r_t 和下一个状态 s_{t+1}

将片段(s_t, a_t, r_t, s_{t+1})存入经验池 D 中

从经验池 D 中随机采样 M 个片段

目标输出 $y_i = \begin{cases} r_i + \gamma Q(s_{i+1}, \pi(s_{i+1}|\theta')|\theta'), \\ r_i, \end{cases}$，如果片段在 i+1 终止则取下式，否则取上式

用随机梯度下降算法最小化损失函数 $L(\theta) = (y_i - Q(s_i, a_i|\theta))^2$

每隔 C 步更新目标 Q 网络，即 θ’ = θ

end

2.3.2 连续动作领域-深度确定性策略梯度网络

正如前面章节所提到的那样，一个参数化的策略对控制问题来说是十分有利的因为它可以运用在连续动作空间。连续动作空间的本质是动作是实数值来描述，比如说控制机械臂的力的大小、机械臂各个关节间的角度值等。深度确定性策略梯度算法是一个动作-值（actor-critic）策略梯度算法，它共有两个网络，一个是 Q 网络，其网络权重为 θ，一个为策略网络 π，其权重为 ϑ。Q 函数的训练目标和深度 Q 网络的一样，唯一不同的是策略 π 现在依赖于 ϑ 而不再是 Q 网络。利用均方差（mean square error）公式，我们可以得到 Q 网络的损失函数：

$$L(\theta) = (r_i + \gamma Q(s_{i+1}, \pi(s_{i+1}|\vartheta')|\theta') - Q(s_i, \pi(s_i|\vartheta)|\theta))^2 \quad (11)$$

此时 Q 网络和策略网络的目标网络分别以一个很小的变化率 μ 来更新，也即 $\theta' = \mu\theta + (1-\mu)\theta'$， $\vartheta' = \mu\vartheta + (1-\mu)\vartheta'$。

策略网络 $\pi(s|\vartheta)$ 通过策略梯度算法来更新：

$$\nabla_{\vartheta} Q(s_i, \pi(s_i|\vartheta)|\theta) = \nabla_a Q(s_i, \pi(s_i|\vartheta)|\theta) \nabla_{\vartheta} \pi(s_i|\vartheta) \tag{12}$$

也就是说，Q 网络的输出就是策略网络的损失函数 $L(\vartheta)$：

$$L(\vartheta) = -Q(s_i, \pi(s_i|\vartheta)|\theta) \tag{13}$$

因为此时动作是连续值，Ornstein Uhlenbeck（OU）噪声 $\Re$ 被加到动作的计算里来确保充分的搜索[27]。同样的，这种异策略方法是可行的因为算法只需要在片段上进行学习。下图表示了深度确定性策略梯度算法的实现流程。

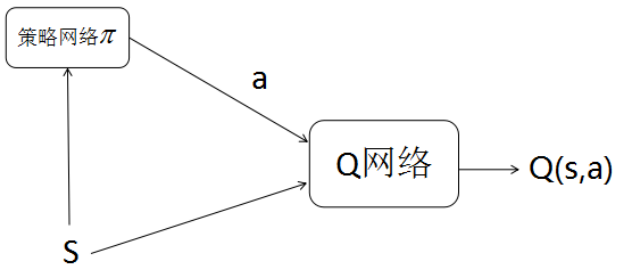


图 2-6：深度确定性策略梯度算法结构示意图

表 2：深度确定性策略梯度算法框架

深度确定性策略梯度算法

初始化经验池 D

初始化策略网络π，权重为ϑ，其目标网络权重为$\vartheta'=\vartheta$

初始化 Q 网络，权重为θ，其目标网络权重为$\theta'=\theta$

得到初始状态 s_1

for t=1 to T **do**

根据$\pi(s_t|\vartheta)+\Re_t$选择动作 a_t

执行动作 a_t，得到奖励 r_t 和下一个状态 s_{t+1}

将片段(s_t，a_t，r_t，s_{t+1})存入经验池 D 中

从经验池 D 中随机采样 M 个片段

目标输出 $y_i=\begin{cases} r_i+\gamma Q(s_{i+1},\pi(s_{i+1}|\vartheta')|\theta'), \\ r_i, \end{cases}$，如果片段在 i+1 终止则取下式，否则取上式

用随机梯度下降算法最小化损失函数 $L(\theta)=(y_i-Q(s_i,a_i|\theta))^2$

用策略梯度算法更新策略网络参数ϑ

更新目标 Q 网络 Q'，$\theta'=\mu\theta+(1-\mu)\theta'$

更新目标策略网络π'，$\vartheta'=\mu\vartheta+(1-\mu)\vartheta'$

end

3 基于优先级的经验回放机制

本章我们将简要介绍传统的经验回放机制算法的内容，然后详细介绍基于优先级的经验回放机制的具体内容，包括评价经验优先级的指标的确定、随机优先级采样的引入、重要性采样系数的加入等。同时，我们也将用一些简明扼要的数学公式来阐明我们提出的算法的合理性和有效性。最后，我们给出基于优先级经验回放机制的 DDPG 算法的完整框架并对算法的一些细节进行了简要阐述。

3.1 传统的经验回放机制

当深度学习算法和强化学习算法融合时，存在一个很大的问题，那就是强化学习的 agent 获得的输入量（观测量）是从环境中连续得到的，因此这些观测量之间存在很强的时序相关性。当将这些观测量直接作为深度网络的输入时，由于深度学习要求输入信息间的相关性要尽可能小，所以这些相关性很强的观测量不满足深度学习对于输入信息的要求。一个有效的解决措施就是使用经验回放机制（experience replay mechanism）。

通常，经验（也叫片段）就是如下形式的元组：(s_t, a_t, r_t, s_{t+1})。经验回放机制的思想就是使用一个很大的内存单元来存储之前从环境中得到的经验，并按照获得的早晚次序顺序存储这些经验。当网络需要输入时，从这个内存单元里随机选择一定数量的经验来进行回放，即作为网络的输入来训练神经网络。当这个内存单元满了的时候，就将最早获得的经验逐个舍弃。这块很大的内存单元又叫经验池（replay buffer）。下图就是经验池添加新经验和舍弃旧经验的示意图：

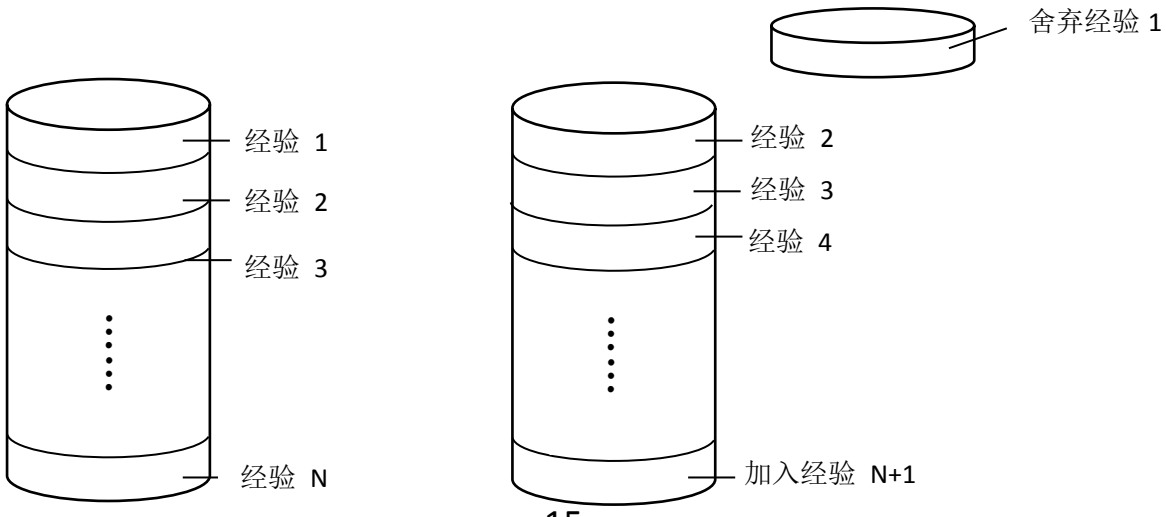


图 3-1：经验回放机制示意图

3.2 基于优先级的经验回放机制

本节主要探究和随机均匀采样相比，按优先级回放经验是怎么样使得经验回放更有效率从而加快 agent 学习过程的。核心思想就是和其它经验相比，一个强化学习 agent 可以从一些经验中学到更多的东西。经验可能或多或少惊讶、冗余或任务相关。一些经验可能不会立即对 agent 有用，但是当 agent 的能力提升时可能就显得非常有用了。传统的经验回放机制让在线学习的 agent 不用再完全以获得的顺序处理经验了。而按优先级的经验回放进一步让 agent 不必以同一频率使用经验了。接下来，我们将具体介绍确定经验优先级的指标、随机优先级算法以及为了消除偏见(bias)的重要性采样系数[28](importance-sampling weight)。

3.2.1 经验优先级的确定

基于优先级的经验回放机制的核心就是选择用来反映经验重要性的指标。很自然地，我们可以想到使用一个 agent 可以从经验学到的知识的多少来作为评价经验重要性的标准。当然，这个指标不太易于实现。一个比较好的的选择是使用经验的 TD-error 作为评价经验有用程度的标准，因为这个数值可以反映它对于 agent 学习过程的帮助程度或是惊讶程度。同时，在许多经典的强化学习算法中如 Q 学习、SARSA，这个数值已经计算出并用于策略网络的更新。

3.2.2 随机优先级采样

既然 TD-error 可以反映经验对于 agent 学习过程的帮助程度或是惊讶程度，那么很自然地我们可以只选择那些 TD-error 大的经验进行回放，也就是贪心优先级回放策略（greedy prioritization)。这样，agent 的学习过程不就会大大加快了吗？理论上可能是这样，不过实际上这种做法存在三个问题。第一，如果采用贪心优先级回放策略，那么只有那些 TD-error 大的经验才会得到回放并更

新它们的 TD-error 数值。这样一来，那些刚开始进入经验池时 TD-error 比较小的经验可能很长时间不会被回放甚至到被舍弃之前都不会被用于策略网络的更新，这种做法无疑会浪费大量宝贵的经验。第二，贪心优先级选择算法只在奖励是确定的情况下才能很好地发挥作用。在我们的测试任务中，奖励是随机的并且函数逼近本身就会带来噪声，此时这个方法就不那么有效了。第三点就是，贪心经验回放只会使得经验池中的很小的一部分经验得到回放并且当使用深度神经网络来训练时偏差只能很缓慢地减小。这种做法使得选择的样本缺少多样性，进而导致网络训练易于过拟合（over-fitting)。

因此，考虑到上述三个问题，我们决定在选择用于回放的经验时加入一定的随机性，即在贪心优先级回放和纯粹的随机经验回放之间进行折中，这种做法被称为随机优先级回放。我们保证经验被回放的概率和它们的 TD-error 大小成正比但是同时也确保那些 TD-error 很小的经验也有机会被回放。具体来说，在选择经验进行策略网络训练时，我们采用基于 rank 优先级的经验回放策略。我们根据经验的 TD-error 绝对值大小来排序从而确定它们的优先级。我们的排序是基于这样的思想的：那些 TD-error 大的经验应该更有可能被回放因为它们对 agent 学习过程的帮助作用最大。我们定义经验池里的 i 个经验被回放的概率是：

$$P(i)=\frac{D_i^{\alpha}}{\sum_k D_k^{\alpha}} \tag{14}$$

其中，$D_i=\frac{1}{rank(i)}$，rank(i)是经验 i 在经验池中根据 TD-error 大小排的次序（rank)。如果参数α被设为 0，那么这就属于随机均匀采样的情况。如果参数α趋向于无穷大，那么这就是贪心优先级算法了。因此，参数α控制优先级使用的程度。

这种随机性的引入还有一个附加的好处。纯粹的随机均匀采样使得用于回放的经验间的时序相关性尽可能小，偏向于选择那些高 TD-error 的经验会不可避免地增大这种相关性。因为一系列不恰当的动作会使得一连串经验拥有很大的 TD-error，这样一连串经验就很可能被选到用于训练策略网络。随机性的引入会使得那些低 TD-error 的经验也有几率被选中，从而减小了经验的时序相关性。

3.2.3 重要性采样系数的加入

基于优先级的经验回放机制会不可避免地改变期望值的收敛分布因为它会倾向于选择那些更加有用的经验，从而改变了状态访问的频率分布。然而，这种变化可能显著地改变训练过程的收敛可能性因为强化学习对于策略网络更新的非偏见性要求（unbiased)。为了尽可能消除这种改变所带来的消极影响，我们在网络权重更新的过程中增加了重要性采样系数（importance-sampling weight)：

$$w_i = \frac{1}{N^{\beta} \cdot P(i)^{\beta}} \tag{15}$$

如果参数β趋向于 0，则意味着对状态访问的频率分布的改变不做任何修正。如果参数β趋向于 1，则意味着十分剧烈地修正优先级采样概率 P(i)。因此，参数β控制这种修正使用的程度。

策略更新的非偏见性要求十分重要，尤其是在网络训练过程的最后阶段，因为在强化学习的场景下训练过程是高度动态的。再者，策略、状态访问频率分布和目标网络在训练过程中一直在变化，导致网络的训练过程易于发散。因此，我们打算使用重要性采样系数，在训练早期轻微修正采样概率 P(i)，并在训练的过程中逐步加大修正力度，最终在训练后期剧烈地修正概率值。换言之，在实验中，我们将参数β的值从 0.5 线性增加到 1。

重要性采样系数还有一个附加的好处。在训练神经网络时，为了保证梯度的估计足够准确，我们通常使用很小的训练步长。然而，在我们的方法中，经验的优先级回放往往使得高 TD-error 的经验被更加频繁地回放。重要性采样系数可以有效减小梯度变化的大小同时消除偏见（bias)，这种做法等同于限制步长的变化幅度。这样，梯度可以被更加准确地估计，从而提高了网络训练收敛的可能性。

以上就是基于优先级的经验回放机制的全部内容。其实，我们所提出的优先级经验回放在其他领域早已熟知。许多神经学研究发现经验回放在啮齿动物的海马体中使用的证据，表明要么在醒着休息或睡觉时先前的经验被回放了。和回报

[29-31]、高 TD-error 相关[32-33]的经验被更加频繁地回放了。众所周知，合理地使用优先更新策略可以使得值函数迭代更加有效。优先级扫描[8,34]选择下一个用于更新的状态，而状态的优先级则由更新执行后状态值的变化来决定。TD-error 提供了一种衡量优先级的办法[35]。我们的方法就是使用一个类似的优先级排序方法，不过是针对无模型的强化学习而不是有模型的计划。并且当从样本中学习一个函数逼近器的时候我们使用随机优先级策略会更加鲁棒。

TD-error 还被用作一种优先机制用于确定选择哪些资源，例如选择探索哪里[36]或者选择什么特征[37-38]。在监督学习中，针对不平衡的数据集，当类别比例已知时有许多技术可以用于处理这个问题，例如再采样、下采样、过采样，并加入集成手段[39]。一个最近的文章在深度强化学习领域引入了一种融合经验回放的再采样技术[40]。该方法将经验分为两块：一块是获得正奖励另一块是获得负奖励的，然后从其中选择一个固定大小的部分用来回放。这仅对那些可以划分正负经验的领域有效（和我们的不同）。并且，Hinton 在 2007 年介绍了一种基于偏差的非均匀采样[41]，并加入重要性采样修正，这使得 MNIST 字体分类任务完成表现提升了三倍。

3.2.4 数学原理解释

在本节，我们将给出基于优先级的确定性策略梯度算法的数学原理。因为我们是通过最小化 Q 网络的损失函数来更新 Q 网络的，所以我们先计算损失函数关于其权重 θ_1 的导数，然后用经验 j 来更新神经网络：

$$\begin{aligned}&\nabla_{\theta_1}(r_t+\gamma Q(st+1,at+1;\theta_1)-Q(st,at;\theta_1))^2\\&=(r_t+\gamma Q(st+1,at+1;\theta_1)-Q(st,at;\theta_1))\nabla_{\theta_1}Q(s,a;\theta_1)\\&=\delta_j\nabla_{\theta_1}Q(s,a;\theta_1)\end{aligned}\qquad(16)$$

基于优先级的经验回放倾向于回放那些高 TD-error 的经验，因此在上式中使用的 δ_j 会比较大，这样可以保证 Q 网络的参数在最陡峭的方向上进行更新。这种做法可以很快地减小 Q 网络的实际输出和期望输出之间的偏差。不过，由于梯度数值的增加，网络权重的变化幅度也会不可避免地增加，这样会使得网络的训练过程易于发散。重要性采样权重的引入可以很好地解决网络权重变化过快的问题，同时提高训练过程的稳定性：

$$w_j \nabla_{\theta_1} L(\theta_1) = w_j \delta_j \nabla_{\theta_1} Q(s,a;\theta_1) \quad (17)$$

从上式，我们很容易看出重要性采样权重的好处。基于优先级的采样导致高 TD-error 的经验被更加频繁地回放，因此 δ_j 会比较大，从而使得梯度变化的估计不太精确。重要性采样权重的加入可以减小梯度变化的幅度因为它的值是小于 1 的。这样，梯度的变化被限制只能向下更新，梯度可以被更加精准地估计，从而稳定了神经网络的训练过程。

Q 网络的稳定、最优的更新方式会同时通过策略梯度影响策略网络的更新过程。因为策略网络的更新可以看成是在 Q 网络所形成的超平面上进行梯度下降算法，所以策略网络更新的稳定性和最优性也得到了保证。从下图，我们可以看出传统的经验回放不可避免地选择那些低 TD-error 的经验进行回放，这样它以一种非常平缓地方式最小化损失函数，却也同时导致多次陷入局部最优，即更新的效率不够高。而基于优先级的经验回放使得更新倾向于相对较大 TD-error 的经验，从而让网络的更新更容易达到全局最优的结果。并且，因为重要性采样权重的引入，它的训练过程也更加稳定。而相反，缺少了重要性采样系数，基于优先级的经验回放会有比较大的训练步长（step size），训练过程也更加容易振荡。因此，重要性采样系数的引入使得损失函数的最小化尽可能达到全局最优，同时也更加稳定。

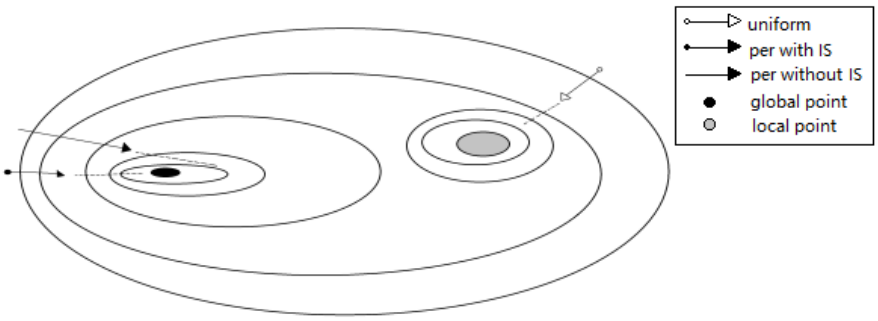


图 3-2：损失函数的最小化过程

3.2.5 完整的算法

在本节，我们将给出完整的基于优先级的深度确定性策略梯度算法，同时算法的一些细节也会被给出。

细节 1：为了最小化由选择哪些经验来回放带来的额外的时间开销，我们采用二叉堆来作为优先级序列的存储结构。这样，找到最高优先级的经验的时间复杂度为 O(1)，更新经验优先级的时间复杂度为 O(logN)。同时，我们用函数 I(.)来描述经验的时间顺序和它相应的优先级之间的映射关系（见下图）。

细节 2：新加入的经验没有一个已知的 TD-error，为了确保每个经验都至少使用一次，我们给这些新加入的经验以最高的优先级，这样经验的使用效率就会极大地提高。

细节 3：为了方便经验的采样过程，经验池被分为 K 个部分（segment）。每个部分都有同样的概率被采到。这些部分的边界可以被事先计算得到，因为优先级为 i 的经验有一个确定的采样概率 P(i)。当 K 为最小批的取值时，在回放经验的过程中每个部分都有一个经验被采到。这样，被回放的经验里总是包含高 TD-error 的经验、中等和低 TD-error 的经验。因此，样本的多样性会大大提高，从而减小了网络过拟合的可能性（经验池的划分算法见下表）。

细节 4：因为对经验池的排序过于频繁会带来很大的时间开销，所以经验池的排序每隔 10^4 才进行一次。在实验中，我们发现这种做法不会给网络的训练带来很大的影响。

Original Order in Replay buffer:	1	2	3	……	N-1	N
Corresponding Priority Order:	I_1	I_2	I_3	……	I_{N-1}	I_N

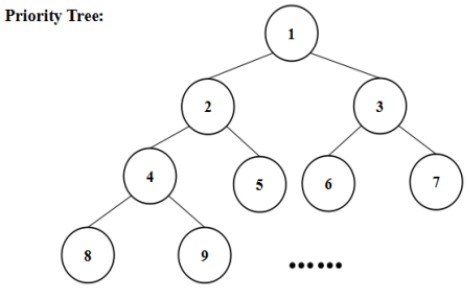


图 3-3：优先级序列的存储结构

表 3：基于优先级的深度强化学习算法

Algorithm 1 DDPG with rank-based prioritization

Initialize critic network $Q(s_t, a_t; \theta_1)$ and actor network $\mu(s_t; \theta_2)$ with random weights θ_1, θ_2
Initialize replay buffer B with size V
Set target network $Q'(s_t, a_t; \theta_1) = Q(s_t, a_t; \theta_1)$, $\mu'(s_t; \theta_2) = \mu(s_t; \theta_2)$
Set priority $D_1 = 1$, exponents $\alpha = 0.7$, $\beta = 0.5$, minibatch K=32
for episode = 1, H **do**
  Initialize a random process $\Re$ used to explore actions thoroughly
  Receive initial state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t; \theta_2) + \Re_t$
    Obtain reward r_t and new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer B and set $D_t = \max_{i<t} D_i$
    if t > V **then**
      for j =1, K **do**
        Sample transition j with probability $P(j) = \frac{D_j^\alpha}{\sum_i D_i^\alpha}$
        Compute importance-sampling weight $W_j = \frac{1}{N^\beta \cdot P(j)^\beta \cdot \max_i w_i}$
        Compute TD-error $\delta_j = r_j + \gamma Q'(s_{j+1}, \mu'(s_{j+1})) - Q(s_j, a_j)$
        Update the priority of transition j according to absolute TD-error $|\delta_j|$
      end for
      Minimize the loss function to update critic network: $L = \frac{1}{K}\sum_i w_i \delta_i^2$
      Use policy gradient to update the actor network:$\nabla_{\theta_2}\mu|_{s_i} \approx \frac{1}{K}\sum_i \nabla_a Q(s_t, a_t; \theta_1)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta_2}\mu(s_t; \theta_2)|_{s_i}$
      Update the target critic network: $Q'(s_t, a_t; \theta_1) = (1-\eta)\cdot Q'(s_t, a_t; \theta_1) + \eta \cdot Q(s_t, a_t; \theta_1)$
      Update the target actor network: $\mu'(s_t; \theta_2) = (1-\eta)\cdot \mu'(s_t; \theta_2) + \eta \cdot \mu(s_t; \theta_2)$
    end if
  end for
end for

表 4：经验池的划分算法

Algorithm 2 Computation of the boundary of K segments

the size of the replay buffer is V, the divided segments of the buffer is K, the exponent of the probability is α
// compute the sum of all experiences' probabilities
sum = 0
for i = 1, V **do**
  sum = sum + $\frac{1}{i^\alpha}$
end for
mark = 1, count = 0
for i = 1, V **do**
  count = count + $\frac{1}{i^\alpha}$
  if count $\geq \frac{mark}{K} \times sum$ **then**
    // the boundary between $mark_{th}$ segment and
    // $(mark+1)_{th}$ segment is i
    mark = mark + 1
  end if
  if mark == K **then**
    break
  end if
end for

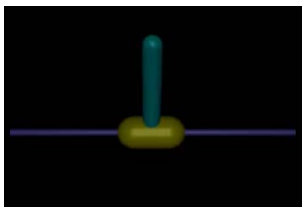
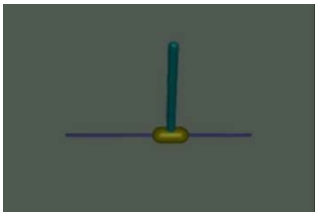
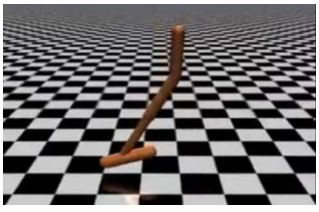
4 实验

当介绍完前面几章的基础知识以后，我们接下来就要做实验来验证基于优先级的经验回放机制到底可以给 agent 的学习过程带来怎样的好处。在下面的几个部分我们将详细介绍实验的环境设置以及 5 个连续控制仿真任务。

首先我们对实验内容做一个简要介绍。所有连续控制的仿真任务模型都是由动力学仿真平台 MuJoCo[42-43]搭建，然后由 OpenAI Gym[44]软件来调用。用来测试的仿真任务总共有 5 个，分别是倒立摆、二级倒立摆、单腿跳、猎豹式奔跑、跑步。其中这 5 个任务的物理学模型维度、动作维度和观测量维度数量如下表：

表 5：仿真任务的模型维度介绍

任务中文名称	英文名称	物理学模型维度	动作维度	观测量维度
倒立摆	Inverted pendulum	4	1	4
二级倒立摆	Inverted double pendulum	6	2	6
单腿跳	hopper	14	4	14
猎豹式奔跑	halfcheetah	18	6	17
跑步	walker	18	6	41

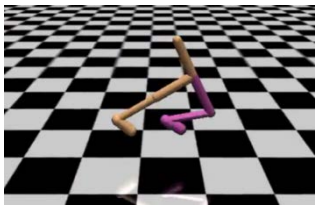


图 4-1：OpenAI Gym 仿真任务示意图.

（从左到右依次是倒立摆、二级倒立摆、单腿跳、猎豹式奔跑和跑步.）

4.1 实验的配置

首先，我们介绍一下实验的基本配置。我们测试的基准算法是深度确定性策略梯度算法，采用的神经网络架构和原论文一致，即采用低维观测信息（如力矩大小、速度、位移等）作为神经网络的输入。实验中总共有两个网络，一个是策略网络 π，一个是 Q 网络（动作-值网络）。这两个网络都包含两个隐层，并且每个隐层神经元个数分别为 400 和 300。对于策略网络，输入为状态观测量，隐层的输出都经过一个整流线性单元（Rectified Linear Unit），输出层为 Tanh 层以保证输出的动作在合理的范围之内。Q 网络是输入为状态观测量和动作，并线性输出 Q 值，以描述在状态 s 下执行动作 a 的价值。其中，输入的动作 a 在第二个隐层才加入。因此，策略网络和 Q 网络最大的不同在于网络的输入和输出（见下图）。为了确保动作搜索的充分性，我们在搜索时加入了 Ornstein-Uhlenbeck（OU）噪声。下表显示了实验中超参（hyperparameter）的取值。

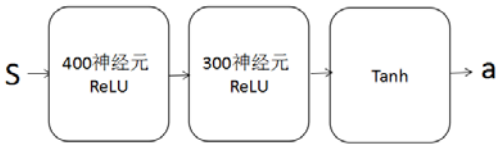


图 4-2：策略网络示意图

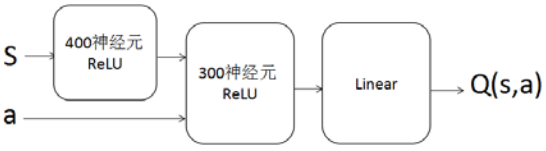


图 4-3：Q 网络示意图

表 6：超参取值表

超参名称	取值
经验池大小	10^4
策略网络学习速率	10^{-4}
Q 网络学习速率	10^{-3}
最小批大小	32
目标网络更新速率	10^{-2}
折扣因子	0.99
OU 噪声参数θ	0.15
OU 噪声参数σ	0.2

4.2 倒立摆仿真实验

我们首先做第一个仿真实验，倒立摆实验。作为经典的连续控制问题，倒立摆的平衡问题是控制领域的经典案例。倒立摆任务的截图如下：

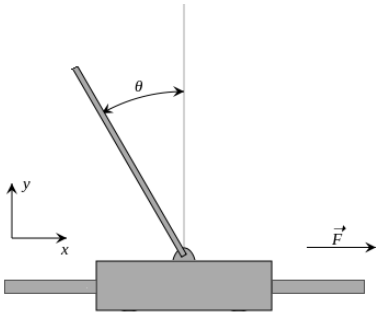


图 4-4：倒立摆的数学模型示意图

我们首先简要介绍一下倒立摆任务的基本内容：一个平衡杆通过一个轴心点固定在一个滑块上，这个滑块可以在一个有限长度的水平木棒上左右移动。agent 可以在这个滑块上施加水平方向上的力使其在木棒上左右移动，而这个 agent 的目标就是保证平衡杆能够尽可能久地保持竖直向上。在这个任务中，agent 可以采取的动作就是施加在滑块上的水平方向的力，agent 可以得到的观测量就是

一个表示倒立摆、滑块的四维向量（平衡杆端点和轴心点的水平坐标和速度）。在实验中，每个时间节点，我们给 agent 一个+1 的奖励当平衡杆和竖直线的夹角小于期望值 θ_{desire}。一个训练回合结束的标志就是角度大于 θ_{desire} 或者滑块移离水平木棒。我们将一个训练回合内的最大训练步数设置为 1000，并且如果 agent 可以在连续 100 个训练回合内获得的奖励值都大于 950.0，那么倒立摆任务被认为已经解决。最大训练步数是 1000 也同样表明在一个训练回合内 agent 可以获得的最大奖励值为 1000。在这个任务以及以下的任务中，一个训练回合包含若干个训练步数。

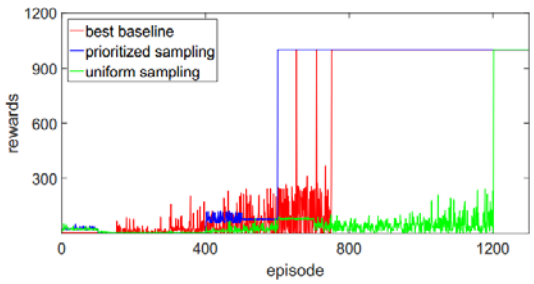


图 4-5：基于优先级的 DDPG 算法和原 DDPG 算法在倒立摆任务上的性能比较

首先，我们在实验中测试三个算法，即基于优先级的 DDPG，原本的 DDPG 以及在 OpenAI Gym 上最好的算法。从上图，我们可以发现就训练的速度而言，基于优先级的 DDPG 很明显地超过了原来的 DDPG 算法。具体地，基于优先级的 agent 花费了 700 个训练回合来完成倒立摆任务的训练而原本的 DDPG 算法花费了 1300 个训练回合才能达到相同的训练效果，也就是连续 100 个训练回合所获奖励值均在 1000。并且，即使和 OpenAI Gym 上最好的算法相比（850 个训练回合），基于优先级的 DDPG 的训练速度也是更快的。另外，从图上我们可以看出，基于优先级的 DDPG 算法的奖励值曲线上的尖刺更加少，这意味着我们的算法的稳定性更好，训练过程更加平稳。这个现象表明，基于优先级的 DDPG 算法可以通过低频地回放低价值的经验并高频回放更加有价值的经验来平滑训练过程，同时提高经验的使用效率。总之，基于优先级的 DDPG 算法通过充分使用经验池里的所有有用经验，同时低频回放那些无用的经验来使得训练过程更加稳定并减少训练时间。

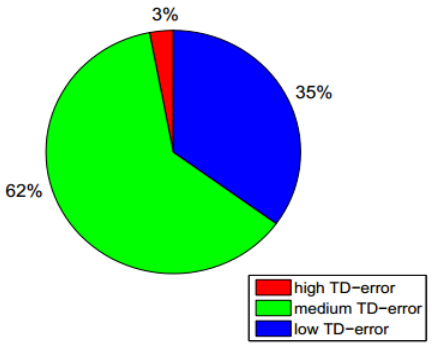


图 4-6：基于优先级的经验回放的采样分布图

其次，我们直观显示基于优先级的 DDPG 采得的样本分布，评价依据就是他们的 TD-error 大小。被选中的经验分为三个部分：高 TD-error 的、中等 TD-error 的和低 TD-error 的。因为 minibatch 的大小为 32，并且经验池的大小为 10000，我们把前 8 块标记为高 TD-error 的经验（第 1 到第 98），最后 8 块标记为低 TD-error 的（第 3544 到第 10000），剩余的经验则为中等 TD-error（第 99 到第 3543，这些块的边界划定是基于分层采样）。从图上，我们明显看出基于优先级的 DDPG 倾向于选择那些 TD-error 值较高的经验因为它们对于 agent 的学习过程更有帮助。基于优先级的 DDPG 也会回放那些比较新的经验因为它们有更大可能性没有被动作-值函数很好地估计。另外，那些低 TD-error 的经验也被采到用于回放，这表明我们所提出的算法采集样本具有多样性的特点。总之，基于优先级的 DDPG 算法不仅能够重点关注那些 TD-error 值较大的经验因为它们有助于学习过程，并且包括那些低 TD-error 的经验以便于增加样本的多样性，防止训练的网络过拟合。不过，令我们吃惊的是，那些低 TD-error 的经验竟然占到选到的样本总数的三分之一。一个可能的原因是当经验池满了的时候，最老的经验会因为新的经验的进入而被舍弃。因为一个循环队列被用于存储经验，那些比较新的经验会有最高的优先级并被放在队列底部。在这种情况下，当这些比较新的经验被选到时，我们会错误地以为它们也是低 TD-error 的经验，从而使得那些被回放的低 TD-error 的经验数量变多。

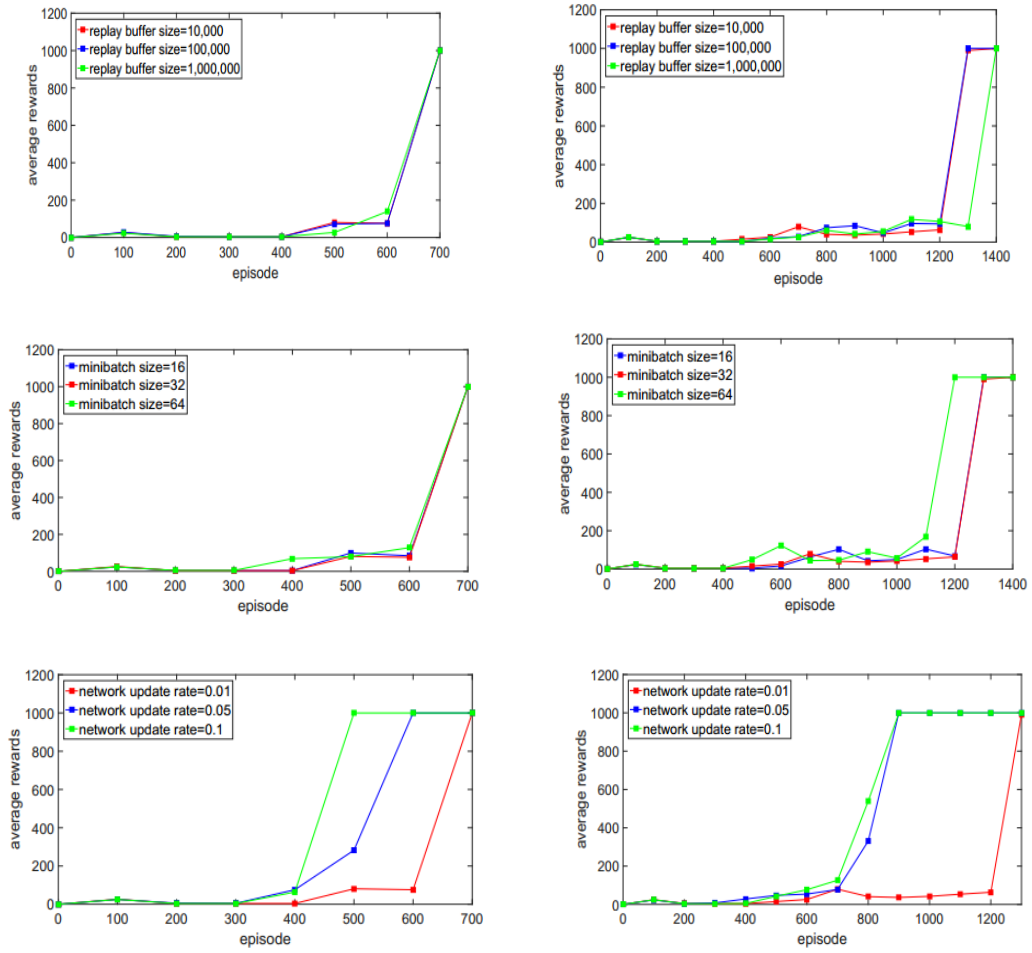


图 4-7：基于优先级的 DDPG 算法和原 DDPG 算法在不同参数下的性能比较

最后，我们改变了实验中一些超参的值（例如经验池的大小、minibatch、目标网络的更新速率）并且测试基于优先级的 DDPG 算法和原本的 DDPG 算法是否足够鲁棒，即算法的性能不会被这些超参的变化影响。另外，两种采样方法的平均奖励被记录以作为算法性能比较的依据因为平均奖励可以更好地反映 agent 的学习趋势。首先，经验池的大小被分别设为 1000000、100000 和 10000。在这种情况下比较两种采样方法的性能变化。从上图，我们发现和原来的 DDPG 算法相比，当经验池的大小变化时，基于优先级的 DDPG 算法展现出更加鲁棒的表现。因为它的平均奖励和总的训练时间依旧分别是 1000 和 700。在这个阶段，当经验池大小增加到 1000000 时，使用均匀采样的 agent 的总训练时间从 1300 增加到 1400 了。一个可能的原因是：因为在均匀采样里经验是随机均匀选出用于回放的。增加经验池的大小无疑会使得那些无用的经验被更加高频地回放来更新网

络，从而使得总的训练时间增加。其次，通过将经验池的大小固定为 10000 并设定 minibatch 的大小分别为 16、32、64，我们比较两种采样方法的性能。从图上，我们惊讶地发现基于优先级的 DDPG 算法的三条平均奖励曲线几乎重合，这意味着基于优先级采样的 agent 此时依旧可以展现稳定的表现。这个现象表明我们算法的鲁棒性之好。同样，采用均匀回放的 agent 的总训练时间增加，并且学习过程有很大的波动。最后，我们将目标网络的更新速率分别设为 0.01、0.05 和 0.1，从而观察两种采样方法的性能变化情况。在实验中，我们发现 agent 的训练时间和这个参数值的大小息息相关。从图上，我们发现当目标网络的更新速率从 0.01 变化到 0.1 时，两个 agent 的训练时间分别减少到 600 和 1000。一个合理的猜测是：这种训练时间的改善归功于训练任务的难度低。具体而言，网络的更新速度的增加会加快网络收敛的速度从而减少总的训练时间。总之，此时基于优先级采样的 agent 依旧保持较快的训练速度。因此，从以上三个超参数值变化的验证实验的比较，我们可以得出一个结论：和均匀采样相比，基于优先级的采样受超参变化影响较小并且更加鲁棒。

4.3 二级倒立摆仿真实验

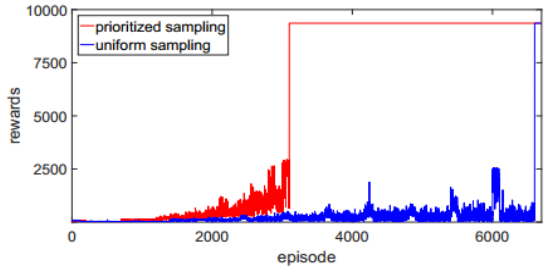


图 4-8：基于优先级的 DDPG 算法和原 DDPG 算法在二级倒立摆任务上的性能比较

接下来，我们将两种采样方法在二级倒立摆实验上进行测试。从上图，我们可以看出基于优先级的 DDPG 算法使得 agent 在 3200 个训练回合完成了二级倒立摆任务而基于随机均匀采样的 agent 则花费了 6700 个训练回合来达到相同的水平（平均奖励值为 9500）。另外，基于优先级的 agent 的奖励值曲线在训练过程中几乎是单调增加的而基于随机均匀采样的 agent 的奖励值曲线在稳定之前有许多上下起伏波动的情况。因此，和上图的分析类似，基于优先级的经验回放机

制通过合理使用过去的经验来使得训练过程更加稳定。

4.4 猎豹式奔跑仿真实验

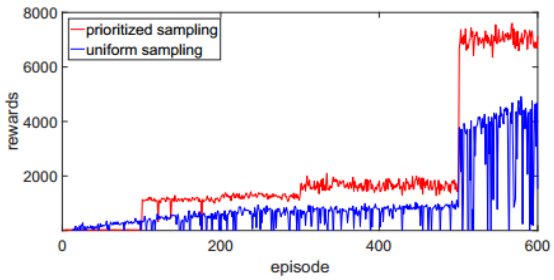


图 4-9：基于优先级的 DDPG 算法和原 DDPG 算法在猎豹式奔跑任务上的性能比较

在猎豹式奔跑仿真实验中，因为在训练初期，动作-值网络（action-value）和 actor 网络都还没有被训练好，因此此时相当于是 agent 在随机胡乱选择动作并得到很少的奖励值。在这种情况下，如果我们刚开始就把一个回合内的最大训练步数设的很大的话，初期 agent 得到的奖励值会是一个很大的负数值从而殃及训练过程。因此，在训练的前 500 个回合我们把最大步数设为 500 之后则调整为 2000。从上图，我们可以看出基于优先级的 DDPG 算法获得的奖励值在该任务上有了很大的提升。基于优先级的 DDPG 算法不仅有一个更加稳定的训练过程，并且最终获得的奖励值也比原本的 DDPG 算法要多很多。这个现象可以这样来解释：因为 TD-error 是反映一个经验对于 agent 的惊讶程度因此高 TD-error 的经验会更加有价值更值得 agent 去学。基于优先级的经验回放机制往往会选择那些高 TD-error 的经验进行回放因此会加速学习过程。基于随机均匀采样的 DDPG 算法会更多地选择那些低 TD-error 的经验进行回放而 agent 只能从这些经验里学到很少的东西。

另外，我们发现在整个训练过程中，均匀采样方法有很多的波动，学习也不平稳尽管它保持增长的趋势。这个现象可以这样来解释：均匀采样没有区别对待经验所以它会不可避免地使得更多老的经验被多次回放并且它们往往会有较小的 TD-error。这些经验不会对训练过程太有帮助因为它们曾经已经被多次回放，而它们的 TD-error 已经调整得比较好了，这意味着它们对 agent 已经不那么有用了。在这种情况下，更多比较新的经验会被回放而它们的 TD-error 的值会比

较大，当这些经验被回放时会减慢甚至殃及 agent 的训练过程。

4.4.1 最大步数改变对实验的影响

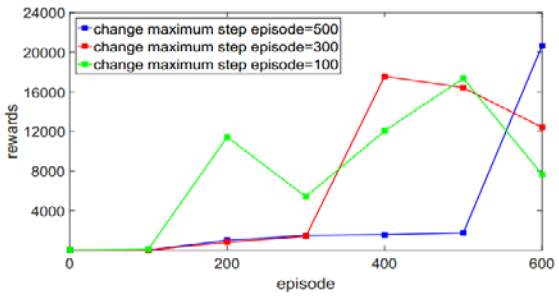


图 4-10：基于优先级的 DDPG 算法在不同最大步长下的性能比较.

在猎豹式奔跑的实验中，我们发现一个训练回合中的最大步数对 agent 的最终表现和神经网络的收敛速度及稳定性有着较大的影响。并且我们在训练初期和后期将最大步数分别设为 500 和 5000，以观察动作-值网络和 actor 网络在这种情况下是否训练好了。具体来说，我们将最大步数分别在前 100、300 和 500 个训练回合设为 500 而在之后设为 5000。这样，我们可以判断最大步数到底以何种程度影响 agent 在猎豹式奔跑任务上的表现。

从上图，我们发现在第 500 个训练回合将最大步数变为 5000 时，基于优先级的 DDPG 算法表现最稳定。这个现象可以这么解释：当经过 500 个训练回合以后，神经网络就已经训练得很好了。当将最大步数改为 5000 时，agent 自然可以很容易表现得更加出色同时因为网络已经训练好，所以其表现也会更加稳定。将最大步数在第 100 个训练回合变为 5000 虽然在前期可以使得 agent 的学习过程更加快速但是却以表现的稳定性为代价。我们在实验中还发现，在第 500 个训练回合以后不断增加最大步数的数值可以使得 agent 获得的奖励值持续增长。这也很好理解，因为策略网络此时已经训练得很好了，在一个回合中给更多的步数来完成任务，agent 无疑会得到更多的奖励值。

4.5 单腿跳仿真实验

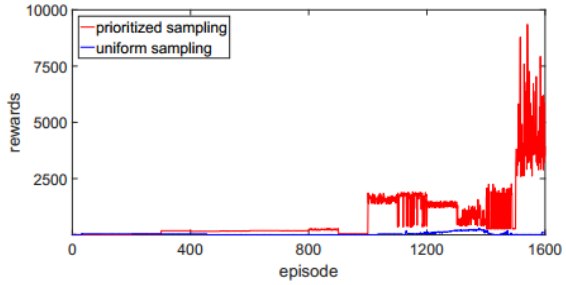


图 4-11：基于优先级的 DDPG 算法和原 DDPG 算法在单腿跳任务上的性能比较

从上图，我们可以发现基于优先级的经验回放机制使得 agent 只用了 1600 个回合就获得平均奖励值为 4400 的表现，而使用传统经验回放机制的 agent 则在同样的训练步数下表现依旧很差。不过，在这个任务中，基于优先级的 DDPG 算法并没有像之前的那些仿真实验那样有很稳定的训练过程。这个现象可以这样来解释：因为和之前实验相比，这个仿真实验中用到的控制量的自由度增加了。在学习过程的初期基于优先级的经验回放机制有可能使得 agent 的表现徘徊在一个局部最优点。当训练过程继续进行时，agent 找到另外一种方法来完成任务。不过这次的方式是较差的一个局部最优，因而导致了学习表现的下滑。这个过程持续直到 agent 找到全局最优的解决方案并获得了最大的奖励值。

4.6 跑步仿真实验

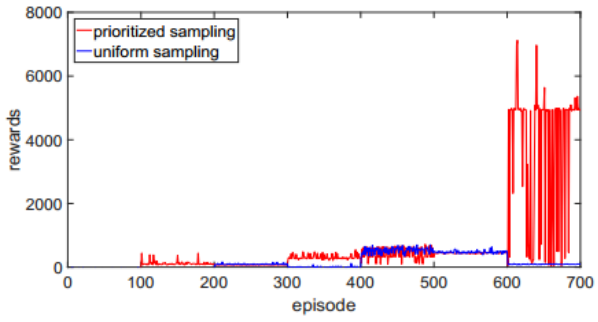


图 4-12：基于优先级的 DDPG 算法和原 DDPG 算法在跑步任务上的性能比较

从上图，我们发现使用优先级回放的 agent 在 700 个训练回合之内就获得了

平均奖励为 6000 的表现而基于随机回放的 agent 表现很差，获得的平均奖励值也就只有 500 左右。然而，使用优先级回放的 agent 的表现同样不是非常稳定。这种表现稳定性上的缺失有可能是因为任务难度的增加。因为 agent 需要更多时间来学习策略，经验池里尝试成功的经验的数目就会少很多，这样就会使得存储在经验池里的经验的价值有所降低。在这种情况下，经验平均质量的下降便会阻碍 agent 的学习过程和表现的稳定性。

为了更加直观地显示出我们算法的优越性，我们将两种经验回放方法在一级倒立摆、二级倒立摆、猎豹式奔跑、单腿跳和跑步这五个仿真任务上的平均奖励值和所需的训练回合数在图上进行了比较。

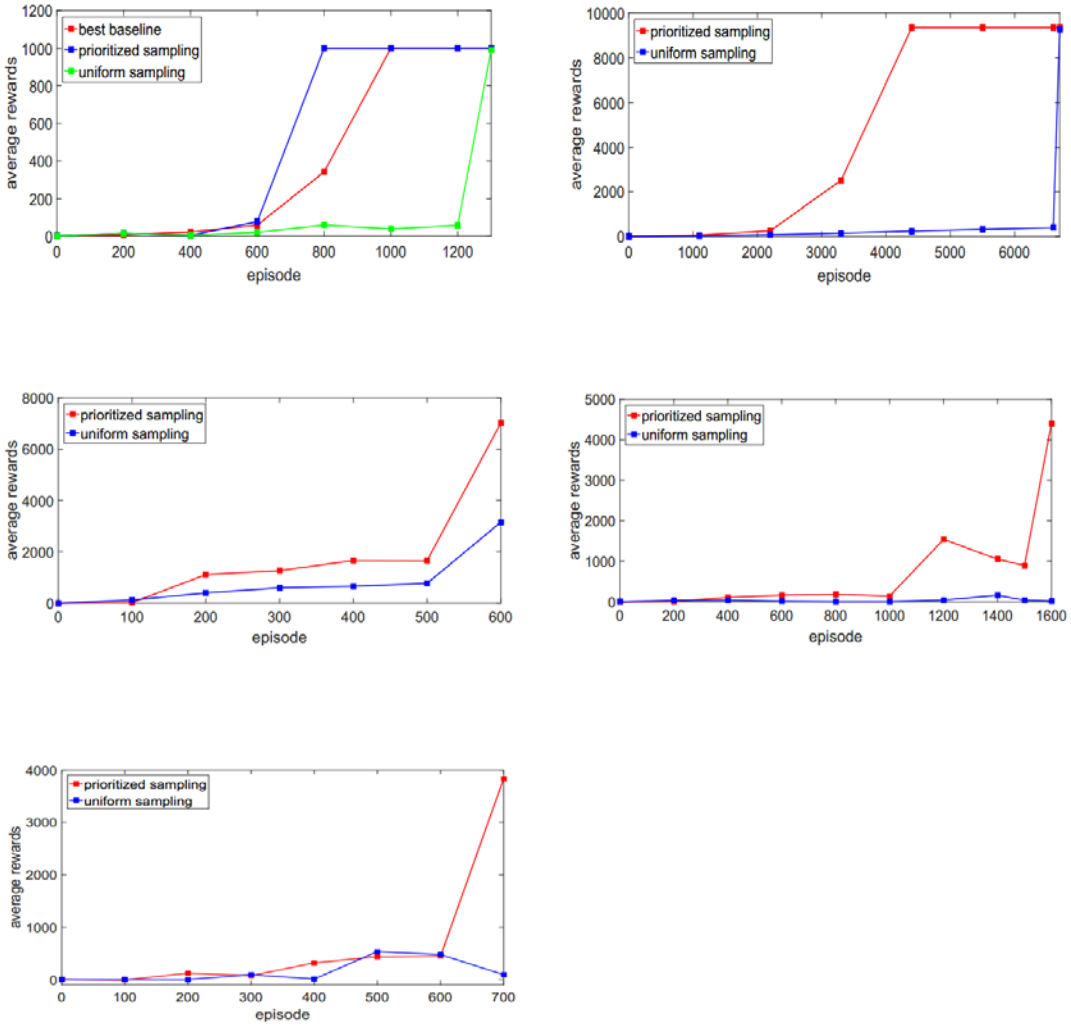


图 4-13：基于优先级的 DDPG 算法和原 DDPG 算法在不同任务上的平均性能比较

4.7 超参变化对实验的影响

在这个实验部分，我们比较两种经验回放方法对超参的大小变化的敏感程度，即测试算法是否足够鲁棒。我们选择了二级倒立摆、猎豹式奔跑、单腿跳和跑步这四个仿真任务，调整的超参为经验池的大小和目标网络更新速率，记录的数据是 agent 获得的平均奖励值。从下图，我们可以很明显看出，因为基于优先级的经验回放机制的 DDPG 算法的奖励值曲线重合度更大，所以其受超参的变化影响较小，也就更加鲁棒。

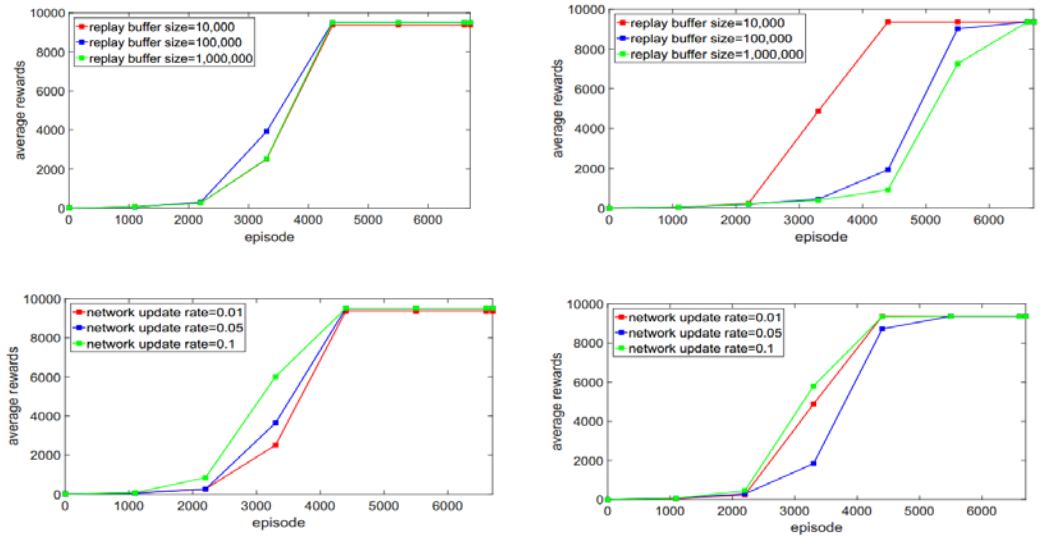
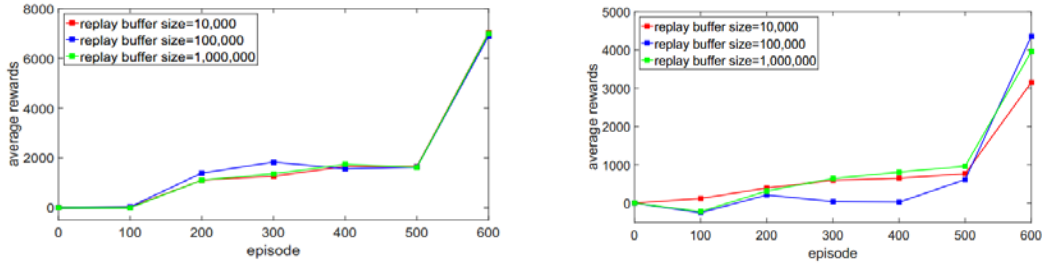


图 4-14：基于优先级的 DDPG 算法和原 DDPG 算法在二级倒立摆任务上的平均性能比较



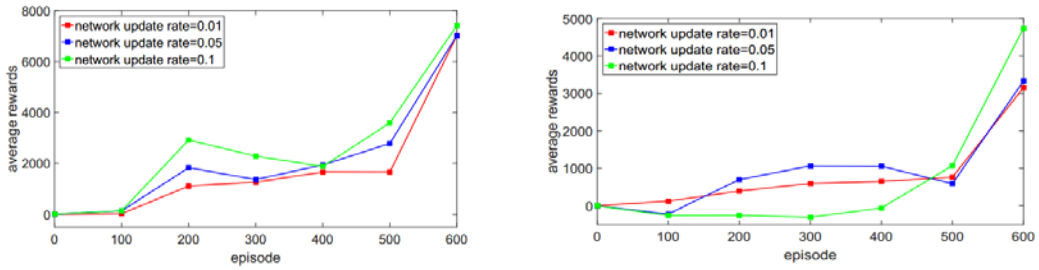


图 4-15：基于优先级的 DDPG 算法和原 DDPG 算法在猎豹式奔跑任务上的平均性能比较

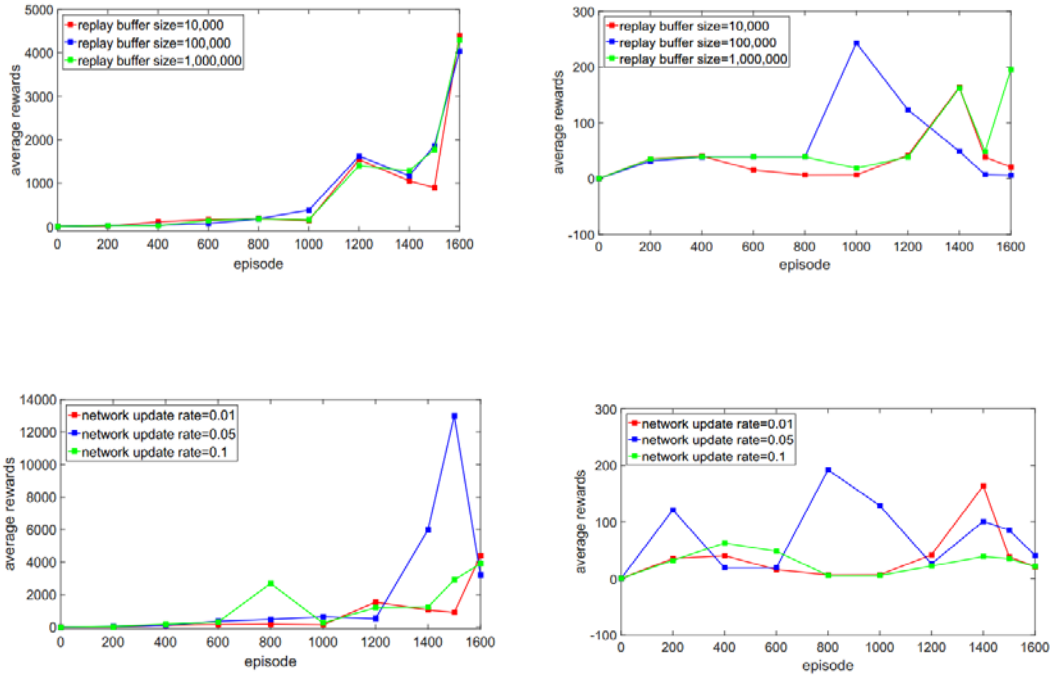
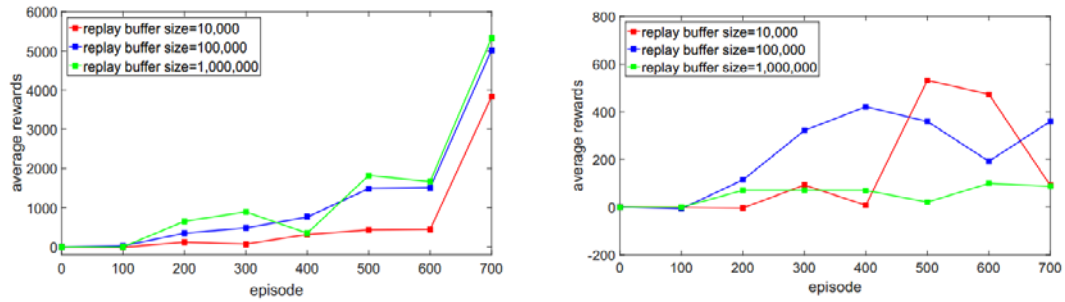


图 4-16：基于优先级的 DDPG 算法和原 DDPG 算法在单腿跳任务上的平均性能比较



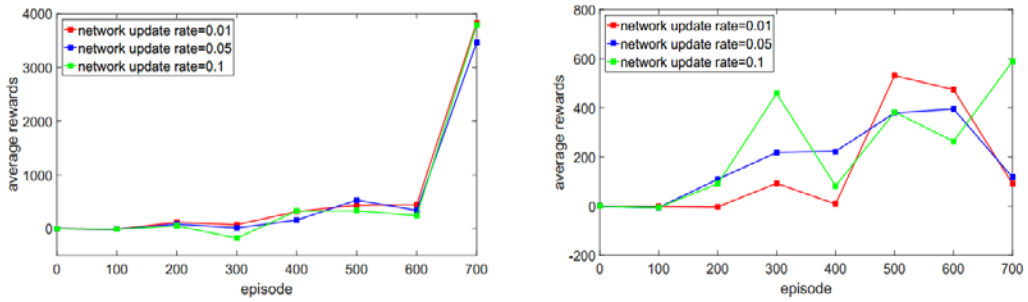


图 4-17：基于优先级的 DDPG 算法和原 DDPG 算法在跑步任务上的平均性能比较

4.8 和其它算法的比较

表 7：四种算法在三个测试任务上的性能比较

（括号旁的是获得的最大平均奖励值，括号里的是所用的训练回合数.）

Task	TRPO	PER-DDPG	TR-c-Q-Prop	DDPG
HalfCheetah	4734($>10^4$)	**21000(500)**	4721(4350)	7490(600)
Hopper	2486(5715)	**4400**(1600)	2957(5945)	2604(**965**)
Walker	3567($>10^4$)	**6900**(4200)	3947(2165)	3626(**2125**)

当下，解决连续控制领域问题的最有效的算法就是 A3C[45]、Q-prop[46]、TRPO[47] 以及 DDPG。从文献中可知，保守式 Q-prop 算法（conservative Q-prop）比激进式 Q-prop 算法(aggressive Q-prop)表现更好。另外，A3C 方法没有使用 OpenAI Gym 因而它的算法表现不具有可比性因为在 MuJoCo 中没有统一的最大平均奖励值和训练回合数的概念。最后，我们在这个部分里只比较 TRPO 算法、保守式 Q-prop 算法（TR-c-Q-prop）、DDPG 算法以及我们提出的基于优先级经验回放的 DDPG 算法（PER-DDPG），测试任务是猎豹式奔跑、单腿跳和跑步三个任务。

从表一，我们可以发现我们提出的基于优先级的 DDPG 算法在单腿跳和跑步任务中获得的奖励值几乎是其它三种算法的两倍。另外，在猎豹式奔跑的任务中，我们的算法和 DDPG 相比，最终获得的奖励值几乎是其三倍，而且我们的算法比 DDPG 所用的训练回合数要更少，这意味着我们的算法是相当成功的。

5 结论与展望

本文提出一种基于优先级的经验回放机制。和传统的经验回放机制相比，基于优先级的经验回放机制能够充分学习那些更加有价值的经验，从而提高经验的使用效率，加快网络的训练过程。实验中，我们将基于优先级的经验回放机制的DDPG 算法和利用传统的经验回放机制的 DDPG 算法进行对比。实验结果表明，和原 DDPG 算法相比，改进的 DDPG 算法不仅可以获得更多的奖励值，而且算法学习过程的稳定性更好，受超参变化的影响较小。在单腿跳、猎豹式奔跑以及跑步这三个任务上，改进的 DDPG 算法比目前最主流的算法的效果都要好。这些结果都有力地证明基于优先级的经验回放机制的有效性。

不过，本文的工作也有一些不足，例如选择的 TD-error 也许不是一个评价经验价值的理想指标，在实验过程中我们也只是考虑选择哪些经验进行回放，并没有考虑舍弃经验的过程。在未来，我希望能够解决这些问题并在更加复杂的任务如 Humanoid 上做一些对比实验。

6 相关的科研成果

[1]Yuenan Hou, Lifeng Liu, Qing Wei, Xudong Xu, Chunlin Chen, “A Novel DDPG Method with Prioritized Experience Replay”, IEEE Systems, Man, and Cybernetics (under review), 2017.

[2]一项发明专利，专利号：CN201610402319.6，发明名称：一种基于深度强化学习的机器人自适应抓取方法.

参考文献

[1] V. Mnih, K. Kavukcuoglu, D. Silver, et al, “Playing Atari with Deep Reinforcement Learning,” Computer Science, pp. 121-131, 2013.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, et al, “Human-level control through deep reinforcement learning,” Nature, pp. 529-533, 2015.

[3] T P. Lillicrap, J J. Hunt, A. Pritzel, et al, “Continuous control with deep reinforcement learning,” Computer Science, 2015.

[4] N. Heess, G. Wayne, D. Silver, et al, “Learning continuous control policies by stochastic value gradients,” Advances in Neural Information Processing Systems, pp. 2944-2952, 2015.

[5] V. Mnih, N. Heess, A. Graves, et al, “Recurrent Models of Visual Attention”, NIPS, pp.2204-2212, 2014.

[6] L J. Lin, “Reinforcement learning for robots using neural networks,”Carnegie Mellon University, 1993.

[7] S. Adam, L. Busoniu, R. Babuska, “Experience Replay for Real-Time Reinforcement Learning Control,” IEEE Transactions on Systems Man and Cybernetics Part C, pp. 201-212, 2012.

[8] A W. Moore, C G. Atkeson, “Prioritized sweeping: Reinforcement learning with less data and less time,” Machine Learning, pp. 103-130, 1993.

[9] T. Schaul, J. Quan, I. Antonoglou, et al, “Prioritized Experience Replay,”IEEE International Conference on Learning Representations, 2016.

[10] W S. McCulloch, W. Pitts, “A logical calculus of the ideas immanent in nervous activity”, the bulletin of mathematical biophysics, 1943.

[11] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain”, Psychological review, 1958.

[12] D E. Rumelhart, G E. Hinton, R J. Williams, “Learning representations by back-propagating errors”, Cognitive modeling, 1988.

[13] G E. Hinton, R R. Salakhutdinov, “Reducing the dimensionality of data with neural networks”, Science, 2006.

[14] A. Mohamed, G. Dahl, G E. Hinton, “Deep belief networks for phone recognition”, Nips workshop on deep learning for speech recognition and related applications, 2009.

[15] T. Mikolov, M. Karafiát, L. Burget, et al, "Recurrent Neural Network Based Language Model", Eleventh Annual Conference of the International Speech Communication Association, 2010.

[16] Y. LeCun, L. Bottou, Y. Bengio, et al, “Gradient-based learning applied to document recognition”, Proceedings of the IEEE, 1998.

[17] A. Krizhevsky, I. Sutskever, G E. Hinton, “Imagenet classification with deep convolutional neural networks”, Advances in neural information processing systems, 2012.

[18] R. Collobert, J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning”, Proceedings of the 25th international conference on Machine learning, 2008.

[19] P. Sermanet, D. Eigen, X. Zhang, et al, “Overfeat: Integrated recognition, localization and detection using convolutional networks”, arXiv preprint arXiv:1312.6229, 2013.

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

[21] C J C H. Watkins, P. Dayan, “Q-learning”, Machine learning, 1992.

[22] C J C H. Watkins, “Learning from delayed rewards”, University of Cambridge, 1989.

[23] J. Peters, S. Schaal, “Natural actor-critic”, Neurocomputing, 2008.

[24] D. Silver, G. Lever, N. Heess, et al, “Deterministic Policy Gradient Algorithms,” International Conference on Machine Learning, pp. 387-395, 2014.

[25] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” arXiv preprint arXiv:1502.03167, 2015.

[26] D E. Goldberg, J H. Holland, “Genetic algorithms and machine learning”, Machine learning, 1988.

[27] G E. Uhlenbeck and L S. Ornstein, “On the Theory of the Brownian Motion”,

Revista Latinoamericana De Microbiologia, 1930.

[28] A R. Mahmood, H V. Hasselt, R S. Sutton, “Weighted importance sampling for off-policy learning with linear function approximation”, International Conference on Neural Information Processing Systems, pp.3014-3022, 2014.

[29] L A. Atherton, D. Dupret, J R. Mellor, “Memory trace replay: the shaping of memory consolidation by neuromodulation”, Trends in neurosciences, 2015.

[30] H F. Olafsdottir, C. Barry, A B. Saleem, et al, “Hippocampal place cells construct reward related sequences through unexplored space”, Elife, 2015.

[31] D J. Foster, M A. Wilson, “Reverse replay of behavioural sequences in hippocampal place cells during the awake state”, Nature, 440(7084):680–683, 2006.

[32] A C. Singer, L M. Frank, “Rewarded outcomes enhance reactivation of experience in the hippocampus”, Neuron, 64(6):910–921, 2009.

[33] C G. McNamara, A. Tejero-Cantero, S. Trouche, et al, “Dopaminergic neurons promote hippocampal reactivation and spatial memory persistence”, Nature neuroscience, 2014.

[34] D. Andre, N. Friedman, R. Parr, “Generalized prioritized sweeping”, In Advances in Neural Information Processing Systems. Citeseer, 1998.

[35] H. van Seijen, R. Sutton, “Planning by prioritized sweeping with small backups”, In Proceedings of The 30th International Conference on Machine Learning, pp. 361–369, 2013.

[36] A. White, J. Modayil, R S. Sutton, “Surprise and curiosity for big data robotics”, In Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.

[37] A. Geramifard, F. Doshi, J. Redding, et al, “Online discovery of feature dependencies”, In Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 881–888, 2011.

[38] Y. Sun, M. Ring, J. Schmidhuber, et al, “Incremental basis construction from temporal difference error”, In Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 481–488, 2011.

[39] M. Galar, A. Fernandez, E. Barrenechea, et al, “A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybridbased approaches”, Systems,

Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 42(4):463–484, 2012.

[40] K. Narasimhan, T. Kulkarni, R. Barzilay, “Language understanding for textbased games using deep reinforcement learning”, In Conference on Empirical Methods in Natural Language Processing (EMNLP), 2015.

[41] G E. Hinton, “To recognize shapes, first learn to generate images”, Progress in brain research, 165:535–547, 2007.

[42] E. Todorov, T. Erez, Y. Tassa, “MuJoCo: A physics engine for modelbased control,” IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5026-5033, 2012.

[43] E. Todorov, T. Erez, Y. Tassa, “Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX”, IEEE International Conference on Robotics and Automation, pp. 4397-4404, 2015.

[44] G. Brockman, V. Cheung, L. Pettersson, et al, “OpenAI Gym,” CoRR, 2016.

[45] V. MnihA P. Badia, M. Mirza, et al, “Asynchronous Methods for Deep Reinforcement Learning,” International Conference on Machine Learning, vol.48, 2016.

[46] S. Gu, T. Lillicrap, Z. Ghahramani, et al, “Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic,” IEEE International Conference on Learning Representations, 2017.

[47] J. Schulman, S. Levine, P. Abbeel, et al, “Trust region policy optimization,” International Conference on Machine Learning, pp. 1889C1897, 2015.

[48] S. Ramstedt, “Deep reinforcement learning for continuous control”, Bachelor Thesis, The Technische Universitat Darmstadt, 2016.

致谢

四年的大学生活转瞬即逝，在这里我向所有指导过、帮助过、关心过我的老师、同学表示衷心的感谢，感谢你们在这四年里给予我的支持和鼓励。同时，我非常感谢指导我毕业设计的陈春林老师，感谢您对我无微不至的关心和指导。跟随您做创新项目和毕业设计的两年期间，我得到的不只是基本的科研素养的训练，还有对问题深入分析的探索精神的磨练和遇到困难百折不挠的坚定决心的打磨，这些品质是我人生宝贵的精神财富。最后，我非常希望自己可以为机器学习这个领域尽一些力量，为人工智能的发展做出一些贡献。