

Email : zxdfgcv@gmail.com

About me : <https://kancheng.github.io/>

0. 作業說明

此報告為人工智慧課程五篇閱讀報告中的第 5 篇 Reinforcement Learning，全報告包含 agent、search、Markov decision process、Bayesian Network、Reinforcement Learning。因為考量自身到對該領域知識的掌握程度不足，全報告採心得與翻譯。

GitHub Project : <https://github.com/kancheng/kan-readpaper-cv-and-ai-in-2021>

1. 原文獻資訊與作者
2. 報告內容心得與講述
3. 原研究文獻

1. 原文獻資訊與作者

Asynchronous Multitask Reinforcement Learning with Dropout for Continuous Control

具有 Dropout 的異步多任務強化學習以實現持續控制

Zilong Jiao;EECS, Syracuse University;Syracuse, NY;zijiao@syr.edu

Jae Oh;EECS, Syracuse University;Syracuse, NY;jcoh@syr.edu

<https://ieeexplore.ieee.org/document/8999228>

Published in: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)

2. 報告內容心得與講述

(1) Motivation 动机

簡單來說該研究的動機在於深度強化學習對於解決複雜任務的樣本效率很低，但近來面對多任務強化學習則因為能夠以提高的樣本效率學習一般策略而受到極大的關注。且在多任務強化學習中，單一 agent 必須依次或同時學習多個相關任務。在此之上研究者想要解決前面因此產生的問題，因而基於 DDPG 演算法，提出了 Asyn-DDPG，提出屬於異步學習多任務策略，以實現同時工作 agent 的連續控制，而且效能在過往 DDPG 前者之上。

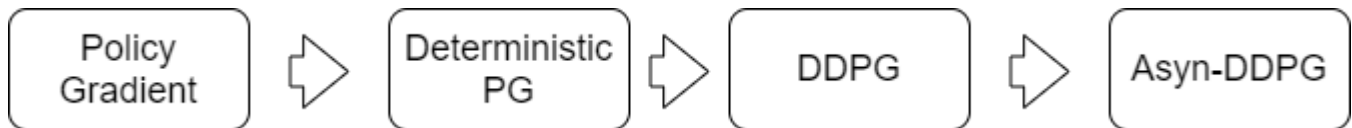
Reference – 閱讀額外知識




<https://zhuanlan.zhihu.com/p/108360386>

<https://arxiv.org/abs/1509.02971>

<https://zhuanlan.zhihu.com/p/107906954>

<https://zh.wikipedia.org/zh-hans/%E5%BC%BA%E5%8C%96%E5%AD%A6%E4%B9%A0>



arXiv  Cornell University  

Computer Science > Machine Learning

arXiv:1509.02971 (cs)

[Submitted on 9 Sep 2015 (v1), last revised 5 Jul 2019 (this version, v6)]

Continuous control with deep reinforcement learning

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

Download PDF

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.

Comments: 10 pages + supplementary

Subjects: **Machine Learning (cs.LG)**; Machine Learning (stat.ML)

Cite as: [arXiv:1509.02971](https://arxiv.org/abs/1509.02971) [cs.LG]

(or [arXiv:1509.02971v6](https://arxiv.org/abs/1509.02971v6) [cs.LG] for this version)

(2) Intuition 直觉

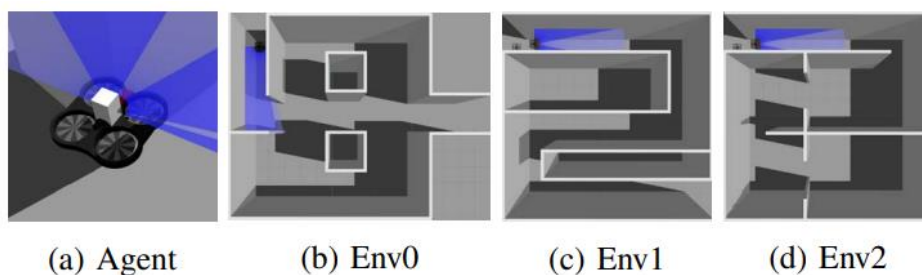


Fig. 3: The local patterns used for training. For each local pattern, a circle is a initial location, and a star is a goal.

Fig. 3: The local patterns used for training.

圖 3：用於訓練的局部模式。

在研究者的過往研究經驗中發現稀疏策略梯度可以顯著減少衝突任務之間的干擾，並使多任務學習有著更加穩定的樣本效率。為了確保為每個任務評估的梯度的稀疏性，**Asyn-DDPG** 將演員和評論家函數都表示為深度神經網絡，並使用 **Dropout** 對其進行正則化。在訓練期間，工作(**agents**)共享 **actor** 和 **critic** 函數，並使用特定於任務的梯度異步優化它們。另外，該策略可以很好地泛化以處理

訓練期間看不見的環境。

(3) Justification 理由

Algorithm 1 A robot applies gradients to shared networks.

```
1: function ASYNUPDATE()
2:   copy  $w^*$  and  $\theta^*$  to  $w$  and  $\theta$ 
3:   while the shared policy hasn't converge do
4:     deploy the robot at a fixed initial location
5:     while an episode is not terminated do
6:       store experience to replay buffer
7:       sample a batch of experience
8:       evaluate policy gradient  $\delta_w$ 
9:       evaluate temporal-difference gradients  $\delta_\theta$ 
10:      apply  $\delta_w$  and  $\delta_\theta$  to  $w^*$  and  $\theta^*$ 
11:      copy  $w^*$  and  $\theta^*$  to  $w$  and  $\theta$ 
12:    end while
13:  end while
14: end function
```

Algorithm 1 A robot applies gradients to shared networks.

演算法 1 機器人將梯度應用於共享網絡。

可以從該研究中的演算法 1，看到這裡所總結的所有 agent 遵循的異步更新程序。 $\mu(\theta^i_t)$ 和 $Q(\omega^i_t)$ 是代理 i 的共享策略和 q 函數的本地副本。在時間 t ，工作 agent i 評估確定性策略梯度和時間差異梯度。

而

- 1) 評估的梯度是稀疏的，代理可以遵循 **Hogwild!** 將梯度異步應用到共享策略和 q 函數的策略。
- 2) 複製機制，在訓練期間，每個工作代理可以將他們在不同任務中學到的知識轉移到其他代理 (agents)。

這允許每個工作代理根據其他 agent 完成的工作改進共享策略，這對於代理將其行為合成為一致的 **meta policy (i.e. the shared policy)** 至關重要。由於梯度評估不依賴於任務特定信息，所有工作代理共同學習的策略可以推廣到不熟悉和可能更複雜的任務。這也是研究者所說的該策略可以很好地泛化以處理訓練期間看不見的環境。

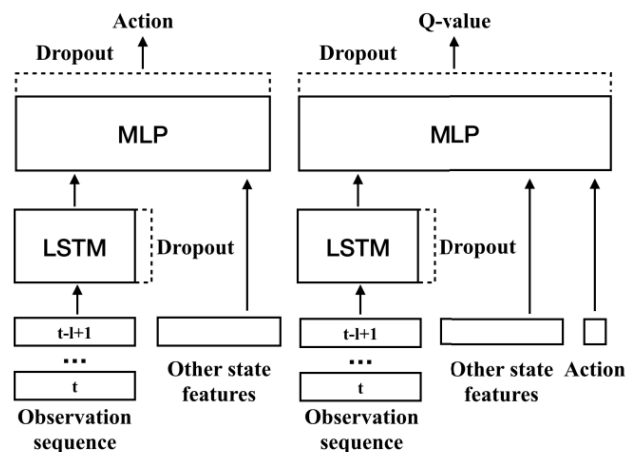


Fig. 2: The shared policy (left) and q-function (right).

Fig. 2: The shared policy (left) and q-function (right). 共享策略（左）和 q 函數（右）。

(4) Framework 框架

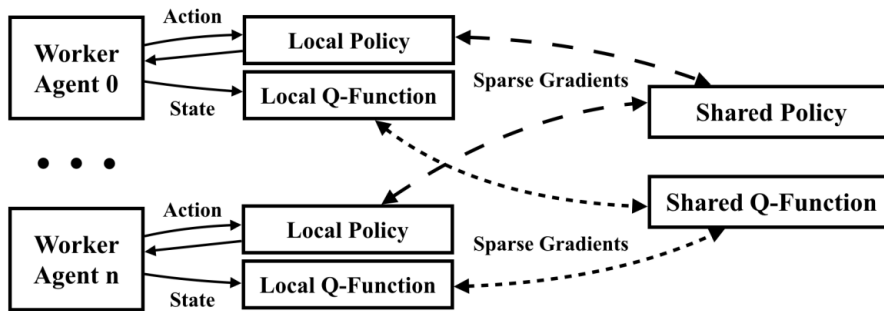


Fig. 1: The overview of Asyn-DDPG

可以看到該研究的異步 DDPG 可以將不同的“退出率”配置應用於其本地策略和 q 函數，且研究者發現讓所有工作 agent 採用相同的退出率配置已經可以讓 Asyn-DDPG 收斂到一個健壯的解決方案。

(5) Result 結果

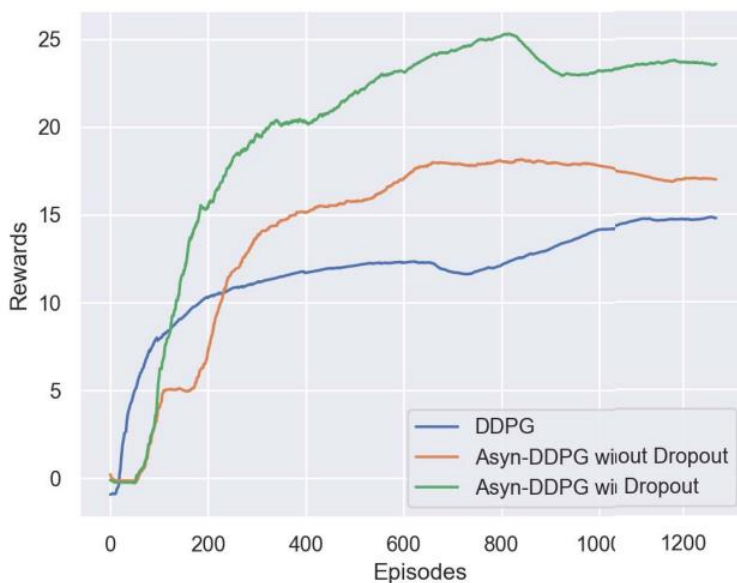


Fig. 4: Learning performance of DDPG and Asyn-DDPG in terms of episodic rewards. Both methods are evaluated in Env3.

在研究文獻中的圖 4，名為 DDPG 和 Asyn-DDPG 在情節獎勵方面的學習性能比較中，可以看到研究者在這兩種方法都在 Env3 中進行了評估，可以看到研究者所提出的方法性能高於 DDPG。

3. 原研究文獻

Abstract 摘要

Deep reinforcement learning is sample inefficient for solving complex tasks.

深度強化學習對於解決複雜任務的樣本效率很低。

Recently, multitask reinforcement learning has received increased attention because of its ability to learn general policies with improved sample efficiency.

最近，多任務強化學習因其能夠以提高的樣本效率學習一般策略而受到越來越多的關注。

In multitask reinforcement learning, a single agent must learn multiple related tasks, either sequentially or simultaneously.

在多任務強化學習中，單個代理必須依次或同時學習多個相關任務。

Based on the DDPG algorithm, this paper presents Asyn-DDPG, which asynchronously learns a multitask policy for continuous control with simultaneous worker agents.

本文基於 DDPG 算法，提出了 Asyn-DDPG，它異步學習多任務策略，以實現同時工作代理的連續控制。

We empirically found that sparse policy gradients can significantly reduce interference among conflicting tasks and make multitask learning more stable and sample efficient.

我們憑經驗發現稀疏策略梯度可以顯著減少衝突任務之間的干擾，並使多任務學習更加穩定和樣本效率。

To ensure the sparsity of gradients evaluated for each task, Asyn-DDPG represents both actor and critic functions as deep neural networks and regularizes them using Dropout.

為了確保為每個任務評估的梯度的稀疏性，Asyn-DDPG 將演員和評論家函數都表示為深度神經網絡，並使用 Dropout 對其進行正則化。

During training, worker agents share the actor and the critic functions, and asynchronously optimize them using task-specific gradients.

在訓練期間，工作代理(agents)共享 actor 和 critic 函數，並使用特定於任務的梯度異步優化它們。

For evaluating Asyn-DDPG, we proposed robotic navigation tasks based on realistically simulated robots and physics-enabled maze-like environments.

為了評估 Asyn-DDPG，我們提出了基於真實模擬機器人和支持物理的迷宮般環境的機器人導航任務。

Although the number of tasks used in our experiment is small, each task is conducted based on a real-world setting and posts a challenging environment.

雖然我們實驗中使用的任務數量很少，但每項任務都是基於現實世界的設置進行的，並發布了一個具有挑戰性的環境。

Through extensive evaluation, we demonstrate that Dropout regularization can effectively stabilize asynchronous learning and enable Asyn-DDPG to outperform DDPG significantly.

通過廣泛的評估，我們證明 Dropout 正則化可以有效地穩定異步學習並使 Asyn-DDPG 的性能顯著優於 DDPG。

Also, Asyn-DDPG was able to learn a multitask policy that can be well generalized for handling environments unseen during training.

此外，Asyn-DDPG 能夠學習一種多任務策略，該策略可以很好地泛化以處理訓練期間看不見的環境。

Index Terms—Deep reinforcement learning, Multitask reinforcement learning, Asynchronous method, Continuous control, Partial observability

I. INTRODUCTION 前言

Recently, deep reinforcement learning has been able to learn policies that can constantly exceed human performance in various task domains, such as Go [16], Atari games [11] and continuous control [9].

最近，深度強化學習已經能夠學習在各種任務領域不斷超越人類表現的策略，例如圍棋 [16]、雅達利遊戲 [11] 和連續控制 [9]。

Even though the results are impressive, the existing work normally trains a specialized policy from scratch for one task at a time, and each task requires training a different policy instance.

儘管結果令人印象深刻，但現有工作通常一次為一項任務從頭開始訓練專門的策略，並且每個任務都需要訓練不同的策略實例。

Learning a specialized policy for complex task in a rich environment usually require tremendous amount of time and agent experience, which make the deep reinforcement learning methods sample inefficient.

在豐富的環境中為複雜任務學習專門的策略通常需要大量的時間和代理經驗，這使得深度強化學習方法樣本效率低下。

To address this issue, researchers in the reinforcement learning community shifted their attention to multitask reinforcement learning which could estimate a general policy by learning a series of related tasks, either sequentially or simultaneously.

為了解決這個問題，強化學習社區的研究人員將注意力轉移到了多任務強化學習上，它可以通過學習一系列相關任務（順序或同時）來估計一般策略。

Compared to single-task reinforcement learning, one would expect that, in multitask reinforcement learning, learning each individual task requires much less data, and combining solutions of multiple tasks

enables a policy to have better asymptotic performance and generalizability.

與單任務強化學習相比，人們會期望，在多任務強化學習中，學習每個單獨的任務需要的數據要少得多，並且多個任務的組合解決方案可以使策略具有更好的漸進性能和泛化性。

Learning simple tasks individually does not make the learning in a multitask setting simpler.

單獨學習簡單的任務不會使多任務環境中的學習變得更簡單。

Instead, it poses at least two stressing issues for learning effective policies.

相反，它為學習有效政策提出了至少兩個緊迫的問題。

First, the processes of learning individual tasks often interfere with each other.

首先，學習單個任務的過程經常相互干擾。

When policy parameters are jointly optimized based on multiple tasks, it is likely that the gradients evaluated in one task can override the gradients evaluated in another task.

當基於多個任務聯合優化策略參數時，一個任務中評估的梯度很可能會覆蓋另一項任務中評估的梯度。

Without special treatments, this would make a multitask reinforcement learning method sample inefficient.

如果沒有特殊處理，這會使多任務強化學習方法樣本效率低下。

Second, a multitask policy can have unbalanced performance on learned tasks.

其次，多任務策略在學習任務上可能具有不平衡的性能。

Since tasks can be learned based on rewards with different scale or distribution, some of the tasks can be more salient than the other during training [7].

由於可以根據不同規模或分佈的獎勵來學習任務，因此在訓練期間，某些任務可能比其他任務更顯著 [7]。

Addressing those issues, recently parallel multitask reinforcement learning [5], [7] has demonstrated remarkable effectiveness in Atari games [11] and DeepMind Lab [2].

為了解決這些問題，最近並行多任務強化學習 [5]、[7] 在 Atari 遊戲 [11] 和 DeepMind Lab [2] 中表現出顯著的有效性。

In those environments, agents operate in discrete action spaces.

在這些環境中，代理在離散的動作空間中運行。

On contrary, multitask reinforcement learning for continuous control is still under-explored.

相反，用於連續控制的多任務強化學習仍在探索中。

Agents with continuous actions are commonly involved in robotic control tasks, such as autonomous driving [14], UAV control [3] and object manipulation [8], [12].

具有連續動作的代理通常涉及機器人控制任務，例如自動駕駛 [14]、無人機控制 [3] 和對像操縱 [8]、[12]。

Unlike game environments [2], [11], robotic control is often constrained on physical factors, e.g. limited sensing ranges, high-dimensional sensor data, and limited acceleration of motion.

與遊戲環境 [2]、[11] 不同，機器人控制通常受到物理因素的限制，例如 有限的傳感範圍、高維傳感器數據和有限的運動加速度。

This paper studies multitask reinforcement learning for continuous control.

本文研究了用於連續控制的多任務強化學習。

In particular, we focus on agents with continuous actions and partial observability.

特別是，我們專注於具有連續動作和部分可觀察性的代理。

Based on Deep Deterministic Policy Gradient (DDPG) algorithm [9], we propose an asynchronous method, Asyn-DDPG, for learning a shared policy and qfunction with multiple simultaneous worker agents.

基於深度確定性策略梯度 (DDPG) 算法 [9]，我們提出了一種異步方法 Asyn-DDPG，用於學習具有多個同時工作代理的共享策略和 qfunction。

Considering partial observability of agents, Asyn-DDPG represents the shared policy and q-function as recurrent neural networks which allow agents to take actions based on a sequence of recent sensor observation.

考慮到代理的部分可觀察性，Asyn-DDPG 將共享策略和 q 函數表示為循環神經網絡，允許代理根據最近的傳感器觀察序列採取行動。

We empirically found that ensuring the sparsity of the gradients applied to the shared policy and q-function can reduce conflicts in learning competing tasks, as well as avoiding unbalanced learning.

我們憑經驗發現，確保應用於共享策略和 q 函數的梯度的稀疏性可以減少學習競爭任務中的衝突，並避免不平衡的學習。

To this end, we regularize the shared policy and q-function using Dropout and ensure the gradients generated through back-propagation to be sparse.

為此，我們使用 Dropout 對共享策略和 q 函數進行正則化，並確保通過反向傳播生成的梯度是稀疏的。

As each agent needs to evaluate its own sparse gradients, applying different Dropout to the same neural network requires synchronization.

由於每個代理 (agent) 都需要評估自己的稀疏梯度(sparse gradients)，因此將不同的 Dropout 應用於同一神經網絡需要同步。

To solve this issue, Asyn-DDPG let each worker agent maintain up-to-date copies of the shared policy and q-function in its own memory and independently apply Dropout regularization to those local copies.

為了解決這個問題，Asyn-DDPG 讓每個工作代理在自己的記憶體中維護共享策略和 q 函數的最新副本，並獨立地將 Dropout 正則化應用於這些本地副本。

During training, each worker agent asynchronously updates the shared policy and q -function using the gradients evaluated based on its regularized local policy and q -function.

在訓練期間，每個工作代理使用基於其正則化本地策略和 q 函數評估的梯度異步更新共享策略和 q 函數。

We evaluate Asyn-DDPG in physic-enabled environments based on robotic simulation.

我們在基於機器人模擬的物理環境中評估 Asyn-DDPG。

In experiments, we let worker agents simultaneously learn different navigation tasks in a small number of maze-like continuous environments.

在實驗中，我們讓工作代理在少量類似迷宮的連續環境中同時學習不同的導航任務。

Experimenting with those environments provides the first step to understand how Dropout regulation affects learning performance of agents in a multitask setting.

在這些環境中進行試驗是了解 Dropout 規則如何影響多任務環境中智能體學習性能的第一步。

It also allows us to analyze the performance of the policy learned by Asyn-DDPG on all those learned tasks in detail.

它還允許我們詳細分析 Asyn-DDPG 在所有這些學習任務上學習的策略的性能。

We demonstrate that Dropout regularization can effectively reduce the interference among competing tasks and enable a learned policy to have balanced performance on individual tasks.

我們證明了 Dropout 正則化可以有效地減少競爭任務之間的干擾，並使學習到的策略在單個任務上具有平衡的性能。

With extensive evaluation, the policy learned by Asyn-DDPG can significantly outperform the specialized policies learned by DDPG in all test environments.

通過廣泛的評估，Asyn-DDPG 學習的策略在所有測試環境中都可以顯著優於 DDPG 學習的專門策略。

In addition, the policy learned by Asyn-DDPG is able to handle more complex navigation tasks that are unseen by agents during training

此外，Asyn-DDPG 學習到的策略能夠處理更複雜的導航任務，這些任務在訓練過程中是代理 (agent) 看不到的

II. RELATED WORK 相關工作

In multitask reinforcement learning, a single agent must learn multiple tasks, either sequentially or simultaneously.

在多任務強化學習中，單個代理必須依次或同時學習多個任務。

In terms of learning multiple tasks sequentially, many work has been studied under the topics of Lifelong Learning [1], [20] and Curriculum Learning [6], [10].

在順序學習多個任務方面，在終身學習 [1]、[20] 和課程學習 [6]、[10] 的主題下已經研究了許多工作。

Recently, simultaneous multitask reinforcement learning has been actively studied, and impressive results have been achieved in discrete environments [5], [7].

最近，同步多任務強化學習得到了積極研究，並在離散環境中取得了令人矚目的成果 [5]、[7]。

On contrary, multitask reinforcement learning in continuous environments are less focused, and recent work on this topics are [4], [22].

相反，連續環境中的多任務強化學習不太集中，最近關於這個主題的工作是 [4]、[22]。

For learning a multitask policy, Deisenroth et al. [4] studied multitask reinforcement learning in the context of robotics for continuous control.

為了學習多任務策略，Deisenroth 等人。 [4] 在機器人技術背景下研究了多任務強化學習以進行連續控制。

In their work, a multitask policy is represented as a function of agent states and tasks, and, at each optimization step, a single agent optimizes the policy using the policy gradients averaged over all tasks.

在他們的工作中，多任務策略被表示為代理狀態和任務的函數，並且在每個優化步驟中，單個代理使用所有任務的平均策略梯度來優化策略。

In contrast, our work does not rely on task identification, instead it enables worker agents to asynchronously optimize the shared policy using regularized sparse gradients.

相比之下，我們的工作不依賴於任務識別，而是使工作代理能夠使用正則化稀疏梯度異步優化共享策略。

This allows our method to be more sample efficient than the aforementioned method.

這使得我們的方法比上述方法更有效。

The closest work to ours is multi-DDPG [22] which learns simple robotic control tasks with multiple DDPG actors.

與我們最接近的工作是多 DDPG [22]，它學習具有多個 DDPG 演員的簡單機器人控制任務。

With a single shared critic, multi-DDPG learns tasks specific actors for each continuous control task.

使用單個共享評論家，多 DDPG 為每個連續控制任務學習任務特定的參與者。

In contrast, our work allows agents to jointly learn a single actor (i.e. a shared policy) to handle all continuous control tasks.

相比之下，我們的工作允許代理聯合學習單個參與者（即共享策略）來處理所有連續控制任務。

This is made possible by using Dropout regularization to resolve conflicts among competing tasks.

這是通過使用 Dropout 正則化解決競爭任務之間的衝突來實現的。

Using regularization to enable multitask learning has been studied in recent literature.

最近的文獻研究了使用正則化來實現多任務學習。

Teh et al. [19] applied γ -discounted KL divergents to task-specific policies, in order to simultaneously distill them into a central policy.

Teh et al. [19] 將 γ -discounted KL divergents 應用於特定任務的策略，以便同時將它們提煉成一個中心策略。

To learn tasks with different reward scales in parallel, Hessel et al. [7] regularize the gradient update of a shared value function by applying PopArt normalization [21] to task-specific state values.

為了並行學習具有不同獎勵等級的任務，Hessel et al. [7] 通過將 PopArt 歸一化 [21] 應用於特定於任務的狀態值來正則化共享值函數的梯度更新。

Comparing to aforementioned methods, Dropout regularization is simple but effective technique for enabling asynchronous multitask reinforcement learning.

與上述方法相比，Dropout 正則化是實現異步多任務強化學習的簡單但有效的技術。

It avoids having need of task specific information (e.g., task IDs) for synthesizing knowledge learned in individual tasks into a meta policy, which sheds light on a better direction for learning a multitask policy.

它避免了需要特定任務信息（例如，任務 ID）來將在單個任務中學到的知識合成到元策略中，這為學習多任務策略指明了更好的方向。

III. BACKGROUND 背景

A. Policy Gradient Methods 策略梯度方法

Policy gradient methods optimize a parameterized policy with respect to its expected reward using gradient decent algorithms.

策略梯度方法使用梯度下降算法優化參數化策略的預期回報。

Let S define a state space of an agent; A be a set of actions the agent can take in each state $s \in S$.

讓 S 定義一個代理的狀態空間； A 是代理在每個狀態 $s \in S$ 中可以採取的一組動作。

μ_θ is a policy with a parameter vector θ .

μ_θ 是具有參數向量 θ 的策略。

When μ_θ is stochastic, the parameter gradients for optimizing its expected reward can be calculated as

當 μ_θ 是隨機的時，用於優化其預期回報的參數梯度可以計算為

$$\nabla_\theta J(\mu_\theta) = \mathbb{E}_{s \sim \rho^{\mu_\theta}, a \sim \mu_\theta} \{ \nabla_\theta \log \mu_\theta(a | s) Q(s, a) \} \quad (1)$$

where ρ^{μ_θ} denotes the probability distribution that an agent visits each $s \in S$ using μ_θ .

其中 ρ^{μ_θ} 表示代理使用 μ_θ 訪問每個 $s \in S$ 的機率分佈。

As a special case of Equation 1, Equation 2 computes the gradients for optimizing a deterministic policy [17].

作為等式 1 的一個特例，等式 2 計算用於優化確定性策略的梯度(the gradients for optimizing a deterministic policy) [17]。

$$\nabla_\theta J(\mu_\theta) = \mathbb{E}_{s \sim \rho^{\mu_\theta}} \{ \nabla_\theta \log \mu_\theta(s) \nabla_a Q(s, a) |_{a=\mu_\theta(s)} \} \quad (2)$$

$\nabla_\theta \log(\mu_\theta(s))$ is a Jacobian matrix where an entry in row i and column j represents the gradient of the i th parameter for the j th action.

$\nabla_\theta \log(\mu_\theta(s))$ 是一個雅可比矩陣，其中第 i 行和第 j 列的條目表示第 j 個動作的第 i 個參數的梯度。

$\nabla_a Q(s, a)$ is a vector of gradients with respect to the Q -function for the action selected by μ_θ in a state.

$\nabla_a Q(s, a)$ 是一個關於由 μ_θ 在狀態中選擇的動作的 Q 函數的梯度向量。

In this paper, we use Q -functions to estimate policy gradients, and alternative estimations of policy gradients can be found in [15].

在本文中，我們使用 Q 函數來估計策略梯度，並且可以在 [15] 中找到策略梯度的替代估計。

If a Q -function (or a value function) is unknown, an agent must fit the unknown Q -function based on its state-action trajectories collected online while calculating policy gradients for optimizing its policy.

如果 Q 函數（或值函數）未知，則代理(agent)必須根據其在線收集的狀態-動作軌跡(state-action trajectories collected)擬合未知 Q 函數，同時計算策略梯度以優化其策略。

A method alternating between fitting a Q -function and optimizing a policy is called an actor-critic method.

在擬合 Q 函數和優化策略之間交替的方法稱為 actor-critic 方法。

B. Sparse Gradients through Dropout Regulation - 通過 Dropout 規則的稀疏梯度

We revisit Dropout [18] in the context of both feed-forward neural networks and recurrent neural networks.

我們在前饋神經網絡和循環神經網絡的背景下重新審視 Dropout [18]。

We present how sparse gradients are obtained when Dropout is applied to both types of neural networks.
我們展示了當 Dropout 應用於兩種類型的神經網絡時如何獲得稀疏梯度。

Dropout in Feed-Forward Neural Network. 前饋神經網路中的 Dropout

Suppose a policy μ_θ is represented as a feed-forward neural network with parameters θ .
假設策略 μ_θ 表示為參數為 θ 的前饋神經網絡。

Let $L = \{1, 2, \dots, l\}$ be the indexes of hidden layers.
令 $L = \{1, 2, \dots, l\}$ 是隱藏層的索引。

z_l and y_l are the input and output of hidden layer l .
 z_l 和 y_l 是隱藏層 l 的輸入和輸出。

In forward operation, dropout is applied to each y_l , s.t.
在前向操作中，dropout 應用於每個 y_l , s.t.

$$\tilde{y}^l = r^l * y^l$$

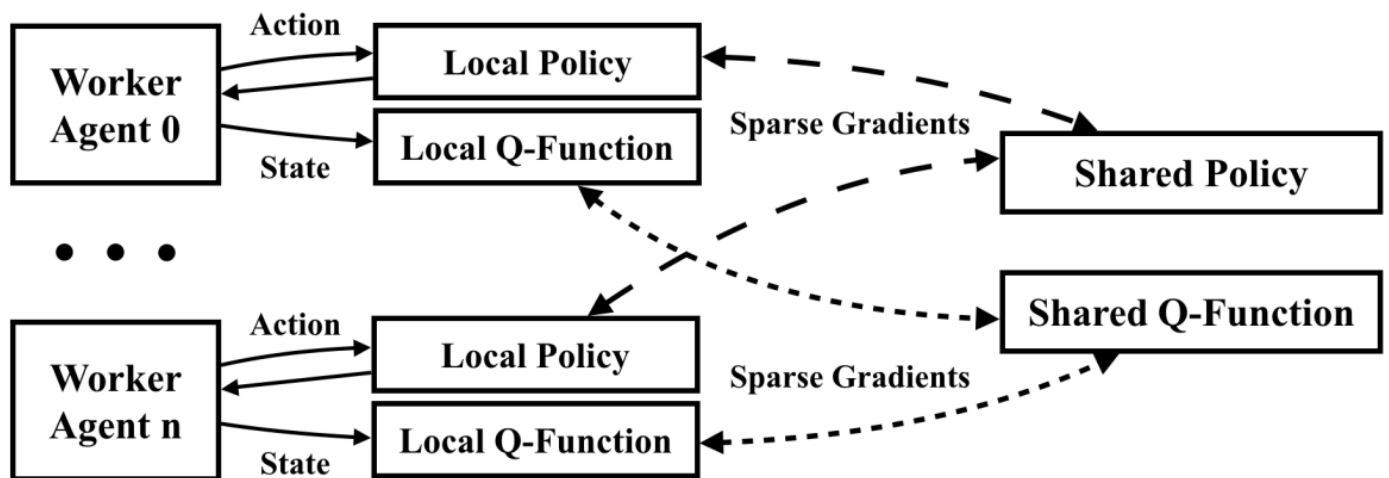


Fig. 1: The overview of Asyn-DDPG

Fig. 1: The overview of Asyn-DDPG

圖 1：Asyn-DDPG 概述

r_l is a vector whose components are independently sampled from Bernoulli distribution with a probability of P being 0, and P is called a dropout rate.

r_l 是一個向量，其分量從伯努利分佈中獨立採樣，概率 P 為 0， P 稱為丟失率。

* denotes element-wise multiplication.

* 表示逐元素乘法。

\tilde{y}^l is input to the hidden layer $l+1$ according to the following equation:

\tilde{y}^l 根據以下等式輸入到隱藏層 $l+1$:

$$z^{l+1} = W^{l+1}\tilde{y}^l + b^{l+1} \quad y^{l+1} = \tau(z^{l+1})$$

where, \tilde{y}^l is a column vector, and τ is non-linear activation function.

其中， \tilde{y}^l 是列向量， τ 是非線性激活函數(non-linear activation function)。

W^{l+1} is a $n \times m$ weight matrix, where n and m are the number of neurons in hidden layers $l+1$ and l .

W^{l+1} 是一個 $n \times m$ 權重矩陣，其中 n 和 m 是隱藏層 $l+1$ 和 l 中的神經元數量。

For computing z^{l+1} , it's equivalent to zero out i th column of W^{l+1} , when i th component of r^l is 0.

對於計算 z^{l+1} ，當 r^l 的第 i 個分量為 0 時，它相當於將 W^{l+1} 的第 i 列歸零。

When the dropout rate P is sufficiently large, all weight matrices in a forward neural network can be sparse.

當 dropout 率 P 足夠大時，前向神經網絡中的所有權重矩陣都可以是稀疏的。

While evaluating gradients for the weight matrices through back-propagation, the entries which were zeroed out in the forward operation are restricted to have gradients of 0.

在通過反向傳播評估權重矩陣的梯度時，在前向操作中清零的條目被限制為梯度為 0。

Therefore, the resulting gradient vector is sparse.

因此，得到的梯度向量是稀疏的。

Dropout in Recurrent Neural Network.

循環神經網絡中的 Dropout。

Taking Long Short Term Memory (LSTM) as an example, we present that Dropout can be applied to a recurrent neural network to produce sparse gradients.

以長短期記憶 (LSTM) 為例，我們提出 Dropout 可以應用於循環神經網絡以產生稀疏梯度。

Let x_t and h_t be the input and the output of LSTM at time t .

設 x_t 和 h_t 是 LSTM 在時間 t 的輸入和輸出。

Dropout is applied to LSTM in the follow way

Dropout 以如下方式應用於 LSTM

$$\tilde{h}_t = r^h * h_t \quad \tilde{x}_t = r^x * x_t$$

r^h and r^x are dropout vectors as what was explained in the feed-forward neural network case.

r^h 和 r^x 是 dropout 向量，正如在前饋神經網絡案例中所解釋的那樣。

t indicates the time step of a input sequence.

t 表示輸入序列的時間步長。

使用 Dropout，LSTM 由以下等式給出：

With Dropout, LSTM is given by the equations below:

$$\begin{aligned}\underline{i} &= \text{sigm}(U_i \tilde{h}_{t-1} + W_i \tilde{x}_t) & \underline{f} &= \text{sigm}(U_f \tilde{h}_{t-1} + W_f \tilde{x}_t) \\ \underline{o} &= \text{sigm}(U_o \tilde{h}_{t-1} + W_o \tilde{x}_t) & \underline{g} &= \text{sigm}(U_g \tilde{h}_{t-1} + W_g \tilde{x}_t) \\ c_t &= \underline{f} * c_{t-1} + \underline{g} * \underline{i} & \underline{h}_t &= \underline{o} * \tanh(c_t)\end{aligned}$$

Let $W = \{W_i, W_f, W_o, W_g\}$ and $U = \{U_i, U_f, U_o, U_g\}$ be weight matrices of a LSTM.

設 $W = \{W_i, W_f, W_o, W_g\}$ 和 $U = \{U_i, U_f, U_o, U_g\}$ 是 LSTM 的權重矩陣。

h_t is a column vector that is the output at time t .

h_t 是一個列向量，它是時間 t 的輸出。

Similar to the feed-forward neural network case, h_t can be calculated by zeroing out i th column of each $W \in W$, if the i th component of \tilde{h}_t is 0.

與前饋神經網絡情況類似，如果 \tilde{h}_t 的第 i 個分量為 0，則可以通過將每個 $W \in W$ 的第 i 列歸零來計算 h_t 。

Similarly, when the i th component of \tilde{x}_t is 0, i th column of $U \in U$ can be zeroed out for computing \tilde{x}_t .

類似地，當 \tilde{x}_t 的第 i 個分量為 0 時，可以將 $U \in U$ 的第 i 列歸零以計算 \tilde{x}_t 。

While evaluating gradients for each $W \in W$ and each $U \in U$, we restrict the entries that are zeroed out to have gradients of 0.

在評估每個 $W \in W$ 和每個 $U \in U$ 的梯度時，我們將清零的條目限制為梯度為 0。

When the dropout probability P is sufficiently large, the resulting gradient vector can be sparse.

當 dropout 概率 P 足夠大時，得到的梯度向量可以是稀疏的。

IV. ASYN-DDPG

Based on DDPG algorithm [9], we propose an asynchronous actor-critic method, Asyn-DDPG, for learning multiple continuous control tasks with worker agents.

基於 DDPG 算法 [9]，我們提出了一種異步 actor-critic 方法 Asyn-DDPG，用於使用工作代理學習多個連續控制任務。

The propose method enables multiple worker agents asynchronously optimize a shared policy and q -function.

該提議方法使多個工作代理異步優化共享策略和 q 函數。

The key to the proposed method is maintaining the sparsity of the gradients applied to the shared policy and q -function.

所提出方法的關鍵是保持應用於共享策略和 q 函數的梯度的稀疏性。

To this end, we let each agent maintain up-to-date copies of the shared policy and q -function in its local memory and apply independent Dropout regularization to those copies.

為此，我們讓每個代理在其本地內存中維護共享策略和 q 函數的最新副本，並對這些副本應用獨立的 Dropout 正則化。

Figure 1 shows the overview of Asyn-DDPG.

圖 1 顯示了 Asyn-DDPG 的概述。

A. Shared Policy with Dropout Regularization - 具有 Dropout 正則化的共享策略

Considering robotic applications in practice, we assume that, at each time step, an agent perceives a high-dimensional feature vector from its surrounding environment through an on-board sensor (e.g. a camera or LiDar).

考慮到機器人在實踐中的應用，我們假設，在每個時間步長，代理(agent)通過車載傳感器（例如相機或激光雷達）從其周圍環境感知高維特徵向量。

The limited sensing capability make the environment where the agent operates partially observable. 有限的感知能力使代理運行的環境部分可觀察。

To overcome the partial observability, in Asyn-DDPG, a state of an agent contains a sequence of sensor observation perceived the past l time steps.

為了克服部分可觀察性，在 Asyn-DDPG 中，代理的狀態包含一系列傳感器觀察到的過去 l 時間步長。

In addition, the state of an agent also contains other information, including locations and velocities. 此外，代理的狀態還包含其他信息，包括位置和速度。

We represent the shared policy and q -function as deep recurrent neural networks with dropout regularization.

我們將共享策略和 q 函數表示為具有 dropout 正則化的深度循環神經網絡。

Figure 2 summarizes their structures.

圖 2 總結了它們的結構。

In Asyn-DDPG, agents independently apply Dropout regularization to the local copies of the share policy and q function.

在 Asyn-DDPG 中，代理獨立地將 Dropout 正則化應用於共享策略和 qfunction 的本地副本。

This allows agents to independently evaluate sparse gradients for optimizing both functions during training. 這允許代理在訓練期間獨立評估稀疏梯度以優化這兩個函數。

Note that agents do not apply Dropout to input of LSTM, since at each optimization step the states input to the local policy and q-function of an agent must always be the same.

請注意，代理不會將 Dropout 應用於 LSTM 的輸入，因為在每個優化步驟中，輸入到本地策略的狀態和代理的 q 函數必須始終相同。

Although agent could apply different configurations of dropout rates to its local policy and q-function, we found that having all worker agents adopt the same configuration of dropout rates can already let Asyn-DDPG converge at a robust solution.

儘管代理可以將不同的退出率配置應用於其本地策略和 q 函數，但我們發現讓所有工作代理採用相同的退出率配置已經可以讓 Asyn-DDPG 收斂到一個健壯的解決方案。

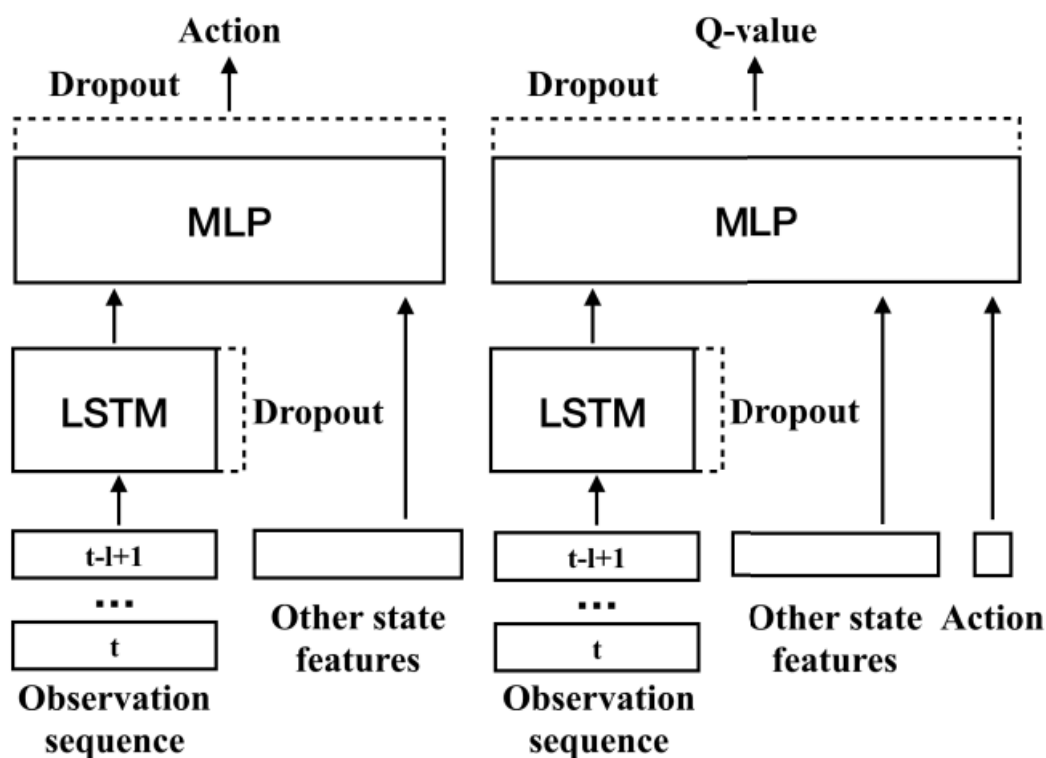


Fig. 2: The shared policy (left) and q-function (right).

Fig. 2: The shared policy (left) and q-function (right).

圖 2：共享策略（左）和 q 函數（右）。

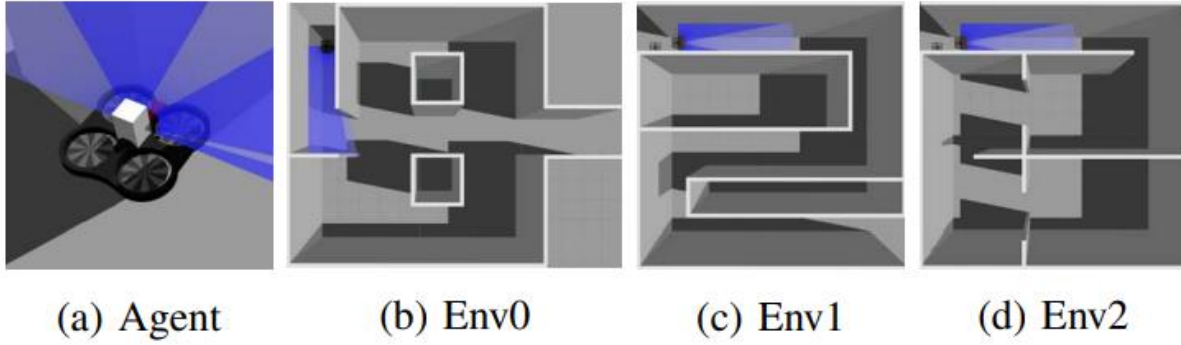


Fig. 3: The local patterns used for training. For each local pattern, a circle is a initial location, and a star is a goal.

Fig. 3: The local patterns used for training.

圖 3：用於訓練的局部模式。

For each local pattern, a circle is a initial location, and a star is a goal.

對於每個局部模式，一個圓圈是一個初始位置，一個星星是一個目標。

Algorithm 1 A robot applies gradients to shared networks.

```

1: function ASYNUPDATE()
2:   copy  $w^*$  and  $\theta^*$  to  $w$  and  $\theta$ 
3:   while the shared policy hasn't converge do
4:     deploy the robot at a fixed initial location
5:     while an episode is not terminated do
6:       store experience to replay buffer
7:       sample a batch of experience
8:       evaluate policy gradient  $\delta_w$ 
9:       evaluate temporal-difference gradients  $\delta_\theta$ 
10:      apply  $\delta_w$  and  $\delta_\theta$  to  $w^*$  and  $\theta^*$ 
11:      copy  $w^*$  and  $\theta^*$  to  $w$  and  $\theta$ 
12:    end while
13:  end while
14: end function

```

Algorithm 1 A robot applies gradients to shared networks.

演算法 1 機器人將梯度應用於共享網絡。

B. Asynchronous Update - 異步更新

Algorithm 1 summarize the asynchronous update procedural followed by all agents.

演算法 1 總結了所有代理遵循的異步更新程序。

$\mu(\theta_{ti})$ and $Q(\omega_{ti})$ are the agent i 's local copies of the shared policy and q-function.

$\mu(\theta_{ti})$ 和 $Q(\omega_{ti})$ 是代理 i 的共享策略和 q 函數的本地副本。

At time t , the worker agent i evaluates Deterministic Policy Gradients and Temporal Difference Gradients.

在時間 t ，工作代理 i 評估確定性策略梯度和時間差異梯度。

Since those evaluated gradients are sparse, agents can follow the Hogwild! Strategy [13] to asynchronously apply their gradients to the shared policy and q-function.

由於那些評估的梯度是稀疏的，代理可以遵循 Hogwild! 將梯度異步應用到共享策略和 q 函數的策略 [13]。

Because of the copying mechanism, during training each worker agent can transfer their knowledge learned in different tasks to the other agents.

由於複製機制，在訓練期間，每個工作代理可以將他們在不同任務中學到的知識轉移到其他代理 (agents)。

This allows each work agent to improve the share policy upon the work done by the other agents, which is essential for the agents to synthesize their behavior into a consistent meta policy (i.e. the shared policy). 這允許每個工作代理根據其他代理完成的工作改進共享策略，這對於代理將其行為合成為一致的元策略（即共享策略）至關重要。

As the gradient evaluation does not depend on task specific information, the policy jointly learned by all worker agents can be generalized to unfamiliar and potentially more complex tasks.

由於梯度評估不依賴於任務特定信息，所有工作代理共同學習的策略可以推廣到不熟悉和可能更複雜的任務。

V. EXPERIMENTS - 實驗

A. Experimental Setup - 實驗裝置

Agents: We simulate worker agents as unmanned aerial vehicles using ROS and Gazebo.

代理：我們使用 ROS 和 Gazebo 將工作代理模擬為無人機。

An worker agent observes an environment through a LiDar sensor, and it scans the area in its front with 360 evenly spaced lasers at a constant rate.

工作人員通過激光雷達傳感器觀察環境，並用 360 個均勻間隔的激光以恆定速率掃描其前方區域。

An agent detects a collision if any laser measures a range less than 0.2m. A state of a worker agent is defined as (o, d, v) .

如果任何激光的測量範圍小於 0.2m，代理就會檢測到碰撞。工作代理的狀態定義為 (o, d, v) 。

o is observation sequence with length of 16. d is the normalized direction from the agent's present location to its goal, and v is the agent's present velocity.

o 是長度為 16 的觀察序列。 d 是從代理當前位置到其目標的歸一化方向， v 是代理當前速度。

In experiments, a worker agent moves on the XY-plane at a constant speed in its heading direction.

在實驗中，工作代理在 XY 平面上以恆定速度沿其前進方向移動。

We define an action of a worker agent to be its rotational velocity, $\gamma \in [-\pi, \pi]$, on the XY-plane.

我們將工作代理的動作定義為它在 XY 平面上的旋轉速度 $\gamma \in [-\pi, \pi]$ 。

Tasks: In experiments, worker agents learn three navigation tasks in maze-like environments, as presented in Figure 3.

任務：在實驗中，工作代理在類似迷宮的環境中學習三個導航任務，如圖 3 所示。

In each environment, a worker agent has a pre-defined initial location and goal.

在每個環境中，工作代理都有一個預定義的初始位置和目標。

It terminate an episode of navigation, if it collides with an obstacle or reaches its goal.

如果它與障礙物碰撞或到達其目標，它會終止導航的一個片段。

In each episode, an worker agent receives rewards given by

在每一集中(In each episode)，一個工人代理(an worker agent)收到 r 給予的獎勵

$$r = \begin{cases} -|\omega| & d'_g \leq d_g \\ d'_g - d_g - |\omega| & d'_g > d_g \\ -1 & \text{close to an obstacle} \\ 1 & \text{reached the goal} \end{cases}$$

close to an obstacle - 接近障礙物

reached the goal - 達到了目標

d_g and d'_g are the Euclidean distances from an agent's previous and present locations to its goal.

d_g 和 d'_g 是從代理之前和現在的位置到其目標的歐幾里德距離。

ω is an agent's present rotational velocity.

ω 是智能體當前的旋轉速度。

The policy jointly learned by all worker agents needs to master navigation tasks in all environments.

所有工作代理共同學習的策略需要掌握所有環境中的導航任務。

B. Learning Performance of Asyn-DDPG - Asyn-DDPG 的學習性能

In the experiment, worker agents uses the same configuration of dropout rates: 0.4 for LSTM and 0.2 for fully connected layers.

在實驗中，工作代理使用相同的丟棄率配置：LSTM 為 0.4，全連接層為 0.2。

For both shared policy and q-function, each fully connected layer has 1024 neurons with ReLu activation functions, and the hidden state of LSTM has the size of 128.

對於共享策略和 q 函數，每個全連接層都有 1024 個具有 ReLu 激活函數的神經元，LSTM 的隱藏狀態大小為 128。

Figure 3 summarizes the learning performance of Asyn-DDPG, in comparison with the standard DDPG and the Asyn-DDPG without dropout regularization.

圖 3 總結了 Asyn-DDPG 與標準 DDPG 和沒有 dropout 正則化的 Asyn-DDPG 的學習性能。

We train the Asyn-DDPG policy with and without dropout regularization in Env0 to Env2 and train the DDPG policy directly in Env3.

我們在 Env0 到 Env2 中訓練有和沒有 dropout 正則化的 Asyn-DDPG 策略，並直接在 Env3 中訓練 DDPG 策略。

For fair comparison, during training we sample the shared policy jointly learned by Asyn-DDPG every 5 seconds (in wall-clock time) and evaluate its performance in Env3.

為了公平比較，在訓練期間，我們每 5 秒（掛鐘時間）對 Asyn-DDPG 聯合學習的共享策略進行採樣，並評估其在 Env3 中的性能。

In the Figure, the reward distribution are the moving average of 100 episodic rewards.

在圖中，獎勵分佈是 100 個情節獎勵的移動平均值。

It shows that the multitask policy learned by Asyn-DDPG achieves better asymptotic performance, compared the single-task policy learned by DDPG.

這表明與 DDPG 學習的單任務策略相比，Asyn-DDPG 學習的多任務策略實現了更好的漸近性能。

To better understand the effectiveness of dropout regularization, we compare the performance of Asyn-DDPG with and without Dropout regularization in Env0 to Env2.

为了更好地理解 dropout 正則化的有效性，我們比較了 Asyn-DDPG 在 Env0 和 Env2 中使用和不使用 Dropout 正則化的性能。

Figure 5 presents the impact of dropout regularization on the Asyn-DDPG.

圖 5 展示了 dropout 正則化對 Asyn-DDPG 的影響。

The results are moving averages of 100 episodic rewards.

結果是 100 個情節獎勵的移動平均值。

Based on the results, Dropout is able to effectively stabilize the performance of Asyn-DDPG on each task. 基於結果，Dropout 能夠有效地穩定 Asyn-DDPG 在每個任務上的性能。

This supports our claim that ensuring sparsity of gradients through Dropout regularization can resolve the interference of learning competing tasks.

這支持了我們的主張，即通過 **Dropout** 正則化確保梯度的稀疏性可以解決學習競爭任務的干擾。

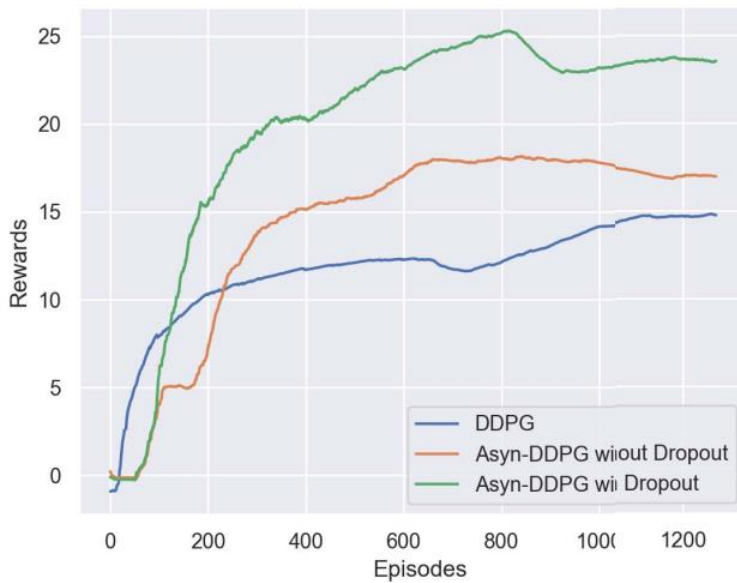


Fig. 4: Learning performance of DDPG and Asyn-DDPG in terms of episodic rewards. Both methods are evaluated in Env3.

Fig. 4: Learning performance of DDPG and Asyn-DDPG in terms of episodic rewards. Both methods are evaluated in Env3.

圖 4：DDPG 和 Asyn-DDPG 在情節獎勵方面的學習性能。
這兩種方法都在 Env3 中進行了評估。

C. Performance Balance of the Learned Multitask Policy - 學習多任務策略的性能平衡

In this section, we evaluate the policy learned by Async-DDPG in two aspects:

在本節中，我們從兩個方面評估 Async-DDPG 學習到的策略：

- 1) if the learned policy performs equally well in training tasks;
1) 如果學習到的策略在訓練任務中表現同樣出色；
- 2) if the learned multitask policy has better performance than task specific policies learned by DDPG.
2) 如果學習的多任務策略比 DDPG 學習的任務特定策略具有更好的性能。

With balanced performance, the multitask policy jointly learned by all worker agents is expected to complete navigation tasks in all training environments.

在平衡性能的情況下，所有工作代理共同學習的多任務策略有望在所有訓練環境中完成導航任務。

For the first aspect of the evaluation, we measure the performance of a policy learned by Async-DDPG in

different concatenations of training environment.

對於評估的第一方面，我們測量了 Async-DDPG 在不同訓練環境串聯中學習的策略的性能。

Those environments are presented in Figure 6a to 6d, together with the trajectories planned by both Async-DDPG and DDPG policies.

這些環境以及 Async-DDPG 和 DDPG 策略規劃的軌跡如圖 6a 到 6d 所示。

We trained those DDPG policies in each of those environments separately, while we only trained one multi-task policy through the Async-DDPG based on Env0 to Env2.

我們分別在這些環境中分別訓練了這些 DDPG 策略，而我們僅通過基於 Env0 到 Env2 的 Async-DDPG 訓練了一個多任務策略。

Table I summarizes the average performance of the trained policies during 50 episodes.

表 I 總結了 50 集訓練策略的平均性能。

According to the results, the multitask policy learned by Async-DDPG can complete the navigation task in all test environments with sufficiently high success rates.

根據結果，Async-DDPG 學習的多任務策略可以在所有測試環境中以足夠高的成功率完成導航任務。

The experiment results provides consistent evidence showing that the multitask policy learned by Async-DDPG has balanced performance in all training tasks.

實驗結果提供了一致的證據，表明 Async-DDPG 學習的多任務策略在所有訓練任務中具有均衡的性能。

Without balanced performance, the learned multitask policy will result in collisions in one of the training environments during navigation.

如果沒有平衡的性能，學習到的多任務策略將導致導航期間在其中一種訓練環境中發生碰撞。

In addition, the multitask policy constantly achieve better performance in all environments, compared to the policies learned by DDPG.

此外，與 DDPG 學習的策略相比，多任務策略在所有環境中不斷獲得更好的性能。

This confirms what we concluded in the previous section:

這證實了我們在上一節中得出的結論：

the policies learned by Async-DDPG has better asymptotic performance, compared to the single-task policies learned by DDPG.

與 DDPG 學習的單任務策略相比，Async-DDPG 學習的策略具有更好的漸近性能。

Env3	Env4	Env5	Env6	Env7	Env8
0.88	0.9	0.9	0.9	0.86	0.92

TABLE I: The success rates of a Asyn-DDPG policy in all test environments based on 50 episodes.

TABLE I: The success rates of a Asyn-DDPG policy in all test environments based on 50 episodes.

表 I：基於 50 集的所有測試環境中 Asyn-DDPG 策略的成功率。

D. Testing Policy Performance in Unfamiliar Tasks - 在不熟悉的任務中測試策略性能

We evaluate the generality of the policy learned by Asyn-DDPG in another two environments which were unseen by worker agents during training.

我們評估了 Asyn-DDPG 在另外兩個環境中學習到的策略的普遍性，而這些環境在訓練期間工人代理是看不到的。

Those environments are presented in Figure 6e and 6f, together with the trajectories planed by the Asyn-DDPG policy.

這些環境與 Asyn-DDPG 策略規劃的軌跡一起顯示在圖 6e 和 6f 中。

Table I shows the success rates of the multi-task policy in all test environments (i.e., Env3 to Env8) based on the average performance of 50 episodes.

表 I 顯示了基於 50 集的平均性能的多任務策略在所有測試環境（即 Env3 到 Env8）中的成功率。

According to the results, the multitask policy learned by Asyn-DDPG can be well generalized to handle unfamiliar tasks.

根據結果，Asyn-DDPG 學習的多任務策略可以很好地泛化以處理不熟悉的任務。

VI. CONCLUSION - 結論

In this paper, we presented an asynchronous multitask reinforcement learning method, Asyn-DDPG.

在本文中，我們提出了一種異步多任務強化學習方法 Asyn-DDPG。

Relying on Dropout regularization, Asyn-DDPG was able to effectively resolve the interference among the processes of learning competing tasks and asynchronously learn a multitask policy with balanced performance and good generalizability.

依靠 Dropout 正則化，Asyn-DDPG 能夠有效解決學習競爭任務過程之間的干擾，異步學習具有均衡性能和良好泛化性的多任務策略。

In experiments, we evaluate Asyn-DDPG in robotic navigation tasks based on realistically simulated robots and physics-enabled environments.

在實驗中，我們基於真實模擬的機器人和支持物理的環境在機器人導航任務中評估 Asyn-DDPG。

Although the number of tasks used in our experiments is small, we designed each individual task based on a real-world setting, and each task posted a more challenging continuous environment, compared to the continuous control tasks studied in the previous multitask reinforcement learning literature [22].

儘管我們實驗中使用的任務數量很少，但我們根據現實世界的設置設計了每個單獨的任務，並且與之前的多任務強化學習文獻中研究的連續控制任務相比，每個任務都發布了更具挑戰性的連續環境 [22]。

With extensive evaluation of the multitask policy learned by Asyn-DDPG, our work provided the first step to understand the effects of Dropout regularization on asynchronous multitask layers.

通過對 Asyn-DDPG 學習的多任務策略的廣泛評估，我們的工作為了解 Dropout 正則化對異步多任務層的影響邁出了第一步。