深度強化學習入門－基於PFRL框架的Lunar Lander環境探索

組長： 林廣哲, 資工二 111010558
組員： 周名初, 資工二 111010561
組員： 林維新, 資工二 111010506
組員： 黃威綸, 資工二 111010514
指導教授： 張珀銀 教授

Abstract

近年來，深度強化學習（Deep Reiforcement Learning，DRL）在人工智慧領域引起了廣泛關注，但它對初學者的學習門檻較高且實作複雜。爲了解決這個問題，出現了一些經過包裝的DRL框架，其中值得關注的一個是PFRL框架。PFRL可以用較少的程式碼解決問題，同時對於神經網路方面提供完整的操作空間，簡單明瞭的程式碼結構使熟悉DRL的開發人員可以快速地轉移到PFRL上，然而，對於初學者來說，這可能是一個挑戰，因爲關於PFRL框架的教學資源相對較少，特別是中文資源。因此，我們藉由PFRL框架，提供了一條針對DRL初學者較平緩的學習路徑，透過OpenAI Gym函式庫提供的LunarLander環境，我們使用DQN、DDQN和DDPG三種常用的經典算法示範PFRL框架的使用。實驗結果表明，我們實現的這三種演算法在得分方面表現良好。具體來說，LunarLander環境的解決分數爲200分，而我們實現的DQN、DDQN和DDPG分別獲得了265、281和269分。這意味著我們提出的解決方案可以有效地在環境中學習並獲得較高的分數表現，對於DRL初學者來說，可以從中學習DRL的完整程式架構與PFRL的API使用。詳細的程式碼與我們的教學可以在https://github.com/KimLinTW/LunarLander_PFRL找到。

Keywords— Reinforcement Learning, DRL, PFRL, Gym, LunarLander

I. 緒論

A. 研究背景

深度強化學習（Deep Reiforcement Learning，DRL）是近年來人工智慧領域中倍受關注的研究焦點，廣泛應用於控制、遊戲[13]、自動駕駛等領域。然而，DRL的開發過程中，需要建立複雜的環境與設計合適的神經網路，對於開發者的能力和時間要求較高。爲了解決這個問題，近年來出現了一些專爲DRL設計的框架，旨在簡化DRL開發過程，減少開發者的負擔，其中PFRL[9]是一個值得關注的框架。

PFRL是一個基於pytorch[12]的DRL框架，優點在於可以用比其他框架更少的程式碼來解決問題，從而減少開發者的負擔，並且可以保有對神經網路的操作空間，以滿足進一步的開發需求。這種簡化DRL開發過程的框架有助於開發者更專注於解決問題本身，而不是繁瑣的DRL環境建立，進一步提高了開發效率和開發品質。

綜上所述，PFRL作爲一個DRL框架，具有簡化DRL開發過程、減少開發者負擔和保有對神經網路的操作空間等優點，對於促進DRL技術的發展和應用具有重要意義。

B. 研究動機

網路上有許多強大的深度強化學習框架，例如PFRL、PARL[10]和Keras[11]等，然而，初入門構建演算法的新手們通常會遇到困難，因爲在其他框架中，撰寫程式的複雜度相對較高（需逾百行程式），這對初學者來說是一個挑戰。相比之下，使用PFRL框架可以簡化程式碼撰寫的過程，並且因爲它需要的設定相對較少，對初學者更加友善。 然而，由於PFRL屬於較新的框架，網路上相關的教學資源非常有限，尤其是中文資源更是寥寥無幾。因此，本研究希望製作一套簡單易懂的PFRL教學，幫助初學者快速了解PFRL框架的使用以及應用於實際問題。

C. 研究問題

1. 如何在PFRL框架下實現不同DRL演算法
2. 如何在PFRL下實現各種DRL智能體並與Lunar Lander環境互動
3. 如何優化PFRL智能體，提高回合獎勵
4. 如何有效地傳達抽象的概念並確保教學品質

D. 研究目的

本研究旨在探討以下目的：

1. 學習並瞭解PFRL框架的使用，包括其架構、格式和演算法的實現。
2. 建立PFRL框架的簡易教學，使其他框架的使用者可以快速地轉換過來。
3. 爲初入演算法領域的新手提供一條完整的學習路徑，透過本研究提供的教學和比較，能夠更好地理解和應用演算法技術。這有助於促進演算法技術的普及和應用，並推動相關領域的發展。
4. 提供一個實際應用PFRL解決MDP問題的具體程式碼

II. 背景知識與文獻探討

A. 深度強化學習

強化學習（Reinforcement Learning，RL）是一種機器學習方法，旨在使智能體（Agent）通過與環境的互動學習最優決策，以達到最大化累積獎勵的目標。在傳統的強化學習方法中，常使用Q表格來存儲和更新狀態-動作對的值函數，以輔助智能體做出決策。

然而，當面對較複雜的環境時，這種基於表格的方法存在一些限制。主要的限制是需要大量的記憶體來存儲Q表格，尤其是當狀態和動作空間非常大時。對於實際問題，如複雜的遊戲或機器人控制，狀態和動作空間往往非常龐大，導致傳統強化學習方法例如: Q-Learning(QL)或Sarsa 難以應用。

另外，QL演算法可以稱作是強化學習最核心的想法，所以在開始嘗試DRL前，完整地理解QL算法的原理是非常重要的，QL算法中的Q值計算公式如下:

$$Q(s,a) = \sum_{s',r} P(s',r|s,a)[r + \gamma \max_{a'} Q(s',a')]$$

其中，$Q(s,a)$代表在狀態s下採取動作a的Q值。 s'代表下一個狀態，r代表獲得的獎勵。 $P(s',r|s,a)$是狀態轉移概率，表示在狀態s下採取動作a後轉移到狀態s'並獲得獎勵r的概率。 γ是折扣因子，用於平衡當前獎勵和未來獎勵的重要性。 $\max_{a'} Q(s',a')$表示在下一個狀態s'下選擇具有最高Q值的動作a'。公式右側的項表示通過選擇具有最大Q值的動作來估計下一個狀態的最大Q值，並乘以折扣因子γ作爲未來獎勵的衰減因子。

這個公式描述了QL算法中Q值的更新規則，它基於當前狀態、動作以及下一個狀態的轉移概率和獎勵來計算Q值。

回到先前提到的存儲Q表格所需的記憶體空間過大的問題，爲了克服這一限制，研究者們引入了深度學習技術，透過數學模型來擬合Q表格，從而產生了深度強化學習（Deep Reinforcement Learning，DRL）。 DRL的運作方式如下圖所示，DRL結合了強化學習和深度學習的優勢，利用神經網路近似Q值函數，從而解決了在複雜環境中需要大量記憶體儲存Q表格的問題。

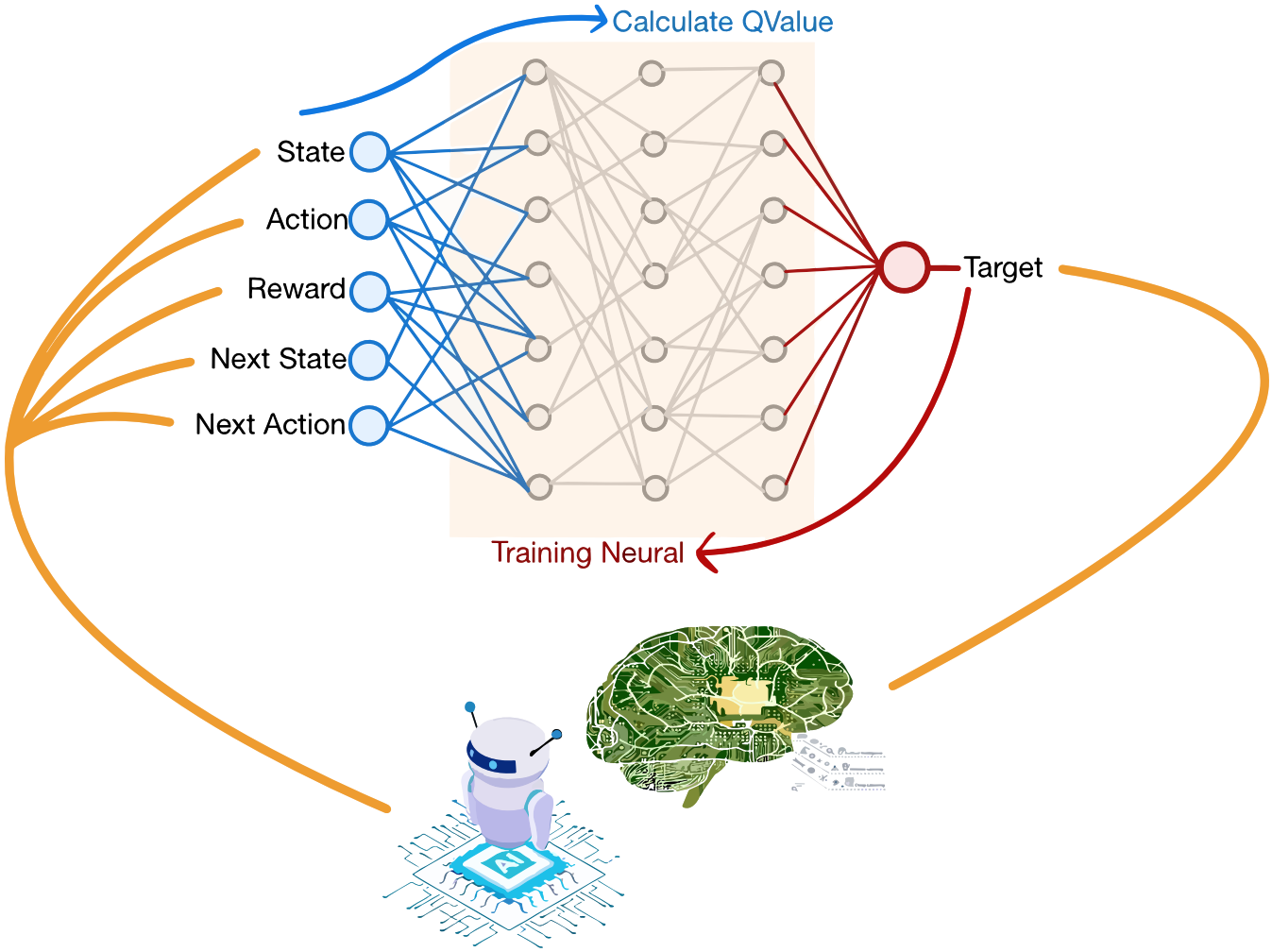

Fig. 1: DRL運作圖

在深度強化學習領域，有幾個經典的算法被廣泛應用和研究。以下將介紹三種經典的深度強化學習算法：DQN、DDQN和DDPG。

1) Deep Q learning (DQN)[1]: DQN是DRL領域中最爲經典的算法之一，許多先進的DRL算法都是由DQN演化而來。例如，Double DQN、Distributional DQN[3]和Rainbow算法都是在DQN的基礎上發展而來的。因此，DQN可以被視爲進入DRL領域的第一步。儘管從傳統的強化學習（QL）過渡到DQN需要具備較高的門檻，但相較於其他DRL算法，DQN算法更加簡單且能夠獲得令人滿意的效果，這也使得DQN成爲當前備受熱議的DRL算法。

DQN除了透過數學模型擬合Q表格外，還使用了2個新技巧：經驗回放（Experience Replay）與目標網路（Target Network）。經驗回放允許智能體從經驗緩衝區中隨機選擇樣本進行訓練，減少樣本相關性，提高訓練效率和穩定性。目標網路則用於降低參數更新的相關性，定期更新目標網路的參數，使其與當前網路保持一致。這些技巧提高了DQN算法的性能和穩定性。

2) Double Deep Q-Networks（DDQN）[2]:: DDQN是對DQN算法的改進，由Hado van Hasselt等人於2015年提出。旨在解決Q值高估問題。在DQN算法中，Q值函數往往會過高估計動作的價值，這可能導致智能體採取次優或不穩定的行動。 爲了解決這個問題，DDQN引入了兩個獨立的神經網路。一個網路用於選擇下一個動作，而另一個網路用於評估該動作的價值。這種設計將動作選擇和價值評估分離開來，減少了Q值的高估問題。通過使用兩個網路並交替更新它們的參數，DDQN能夠更準確地估計動作的眞實價值，提高在複雜任務上的性能和穩定性。

3) Deep Deterministic Policy Gradient（DDPG）[4]:: DDPG算法用於解決連續動作空間下的強化學習問題。 結合了深度神經網路和確定性策略梯度方法，能夠處理高維狀態和動作空間，並在連續控制任務中取得了良好的性能。

DDPG算法包括兩個主要組成部分(圖2)：表演者（Actor）和評論家（Critic）。表演者網路通過輸入當前狀態來生成連續動作，它的目標是學習一個最優策略來最大化長期累積獎勵。評論家網路是一個Q值函數估計器，用於評估表演者網路輸出的動作的價值。通過最小化表演者生成動作和評論家估計的動作價值之間的誤差，DDPG使用確定性策略梯度算法來更新表演者和評論家網路的參數。

與其他基於價值函數的方法相比，DDPG具有以下幾個特點：首先，它使用了深度神經網路來逼近策略和Q值函數，使得它能夠處理高維狀態和動作空間；其次，DDPG是一個off-policy算法，可以利用經驗回放技術來提高訓練的穩定性和樣本效率；DDPG算法具有探索和利用的平衡，通過添加噪聲來探索新的動作，同時利用表演者網路生成的動作來實現更好的策略優化。

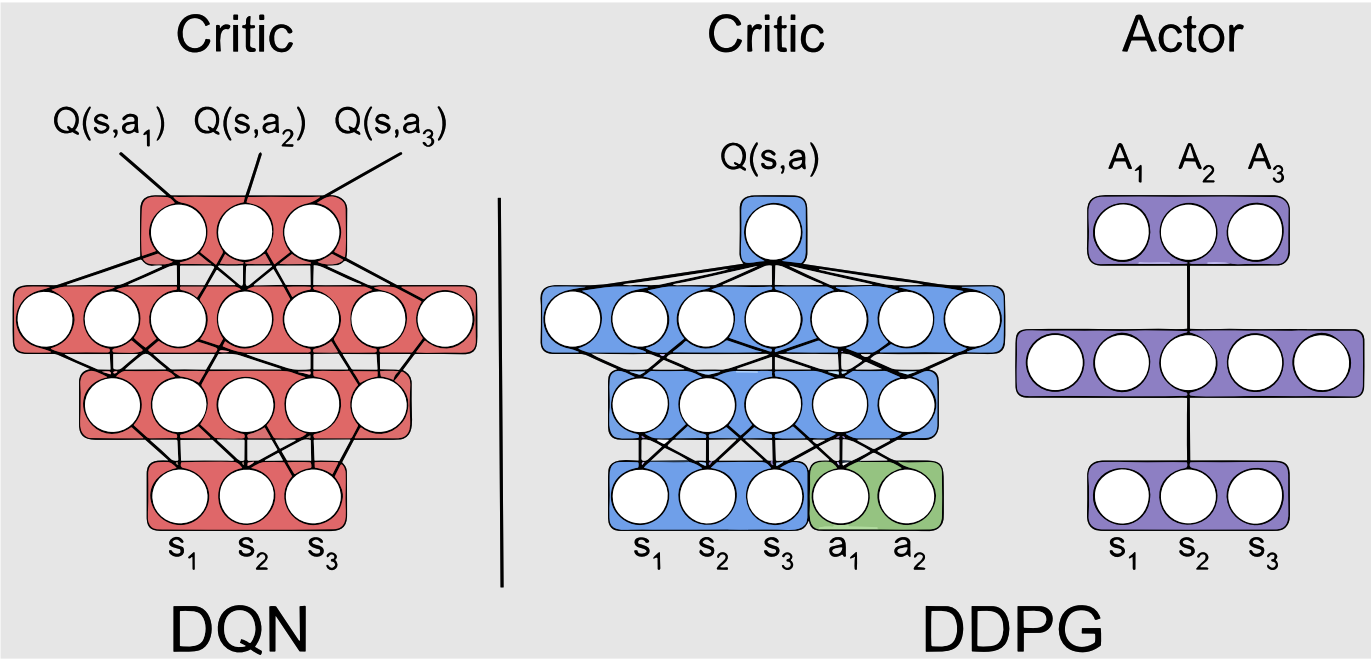

Fig. 2: DQN vs DDPG

B. 現有的*DRL*學習資源

在網路上，有許多DRL的學習資源，包括影片、部落格和筆記等。其中，一些教學影片使用

了實際的DRL框架（例如PARL[10]），進行教學和示範。然而，這些教學所使用的框架通常對於DRL初學者來說並不太友善。

爲了學習DRL的基礎技術，包括智能體的探索方式、經驗池和神經網路，需要撰寫大量的程式碼來實現，或者使用現有的DRL框架。然而，這些框架可能會有一些問題。有些框架過於複雜難以上手，需要花費大量時間去理解和應用；而有些框架則過於簡單，無法應付大多數情況，因爲它們在設計上犧牲了一些參數或神經網路結構的彈性，只能應對特定的場景。因此，採用這種方法的學習步驟往往需要重新學習其他具有較大彈性的DRL框架。

C. PyTorch-based Reinforcement Learning(PFRL)[9]

PFRL是一個基於PyTorch的強化學習框架，它具有以下特點：

1. 簡單易用：PFRL提供了簡潔而直觀的API，使得用户可以輕鬆地定義、訓練和評估強化學習模型。它的設計目標是讓用户專注於算法的實現和問題的建模，而不必過多關注底層細節。

2. 靈活性：PFRL提供了豐富的模組和組件，可以靈活地構建和定制各種強化學習算法。使用者可根據需求選擇不同的網路結構、優化器、經驗回放等組件，以實現特定的算法或進行實驗。

3. 多樣性的算法支持：PFRL實現了許多經典和最新的強化學習算法，如下圖所示，包括DQN、DDQN、DDPG、A3C、PPO、SAC等。這些算法涵蓋了DQN、Policy Gradient、actor-critic等，使用者可以根據問題的特點選擇合適的算法進行使用和擴展。

與其他強化學習框架相比，PFRL在實現的算法數量上具有一定的優勢。它已經實現了多種經典算法和最新算法，涵蓋了強化學習領域的主要範式和方法。這使得用户可以更方便地選擇合適的算法，並且可以根據需要進行擴展和定制。

TABLE I: PFRL實現演算法種類

Algorithm	Discrete Action	Continous Action	Recurrent Model	Batch Training	CPU Async Training	Pretrained models*
DQN (including DoubleDQN etc.)	✓	✓ (NAF)	✓	✓	x	✓
Categorical DQN	✓	x	✓	✓	x	x
Rainbow	✓	x	✓	✓	x	✓
IQN	✓	x	✓	✓	x	✓
DDPG	x	✓	x	✓	x	✓
A3C	✓	✓	✓	✓ (A2C)	✓	✓
ACER	✓	✓	✓	x	✓	x
PPO	✓	✓	✓	✓	x	✓
TRPO	✓	✓	✓	✓	x	✓
TD3	x	✓	x	✓	x	✓
SAC	x	✓	x	✓	x	✓

D. LunarLander[5]

LunarLander是一個常用的強化學習測試環境，用於評估智能體在複雜動力學環境中的控制能力。該環境模擬了一艘登月艙在月球表面上的著陸任務。

在LunarLander環境中，智能體需要學會控制登月艙，使其在限制時間內盡可能平穩地降落在目標區域。智能體可以選擇四個離散的動作：向左推力、向右推力、向下推力或不進行推力。通過選擇合適的動作，智能體需要避免失去控制、碰撞或不正確的著陸姿態。

該環境提供了豐富的狀態信息，包括登月艙的位置、速度、角度和角速度等。智能體通過與環境的交互來學習合適的動作策略，以最大化累積獎勵。

LunarLander的複雜動力學和非線性特性使其成爲一個具有挑戰性的強化學習任務。智能體需要克服重力、推力和慣性等物理因素，並準確感知和響應環境的變化，以實現穩定而有效的著陸。

在研究和開發強化學習算法時，LunarLander作爲一個標準測試環境被廣泛應用。研究人員可以使用不同的算法和技術來訓練智能體，在LunarLander環境中評估其性能和控制能力。通過與其他算法的比較和性能分析，研究人員可以深入了解不同算法在複雜任務上的優劣，並推動強化學習領域的發展。

因此，LunarLander提供了一個具有挑戰性和實用性的測試平台，用於研究強化學習算法的效果、改進和比較，並促進智能體在複雜動態環境中的控制能力的進步。

LunarLander具體的規則與計分方式如下: 在每一步之後，遊戲會給予一個獎勵，而整個回合（episode）的總獎勵則是該回合中所有步驟獎勵的總和。 對於每一步，獎勵的計算方式如下：

靠近或遠離目標位置：太空船距離目標位置越近，獎勵越高；距離越遠，獎勵越低。
太空船的速度：太空船速度越慢，獎勵越高；速度越快，獎勵越低。
太空船的傾斜角度：太空船的傾斜角度越大，獎勵越低。
腳部接觸地面：每個腳部與地面接觸，會增加10分的獎勵。
側向引擎的使用：每幀開啓側向引擎會減少0.03分的獎勵。
主引擎的使用：每幀開啓主引擎會減少0.3分的獎勵。
在回合結束時，如果太空船發生墜毀，會額外扣除100分的獎勵；如果太空船安全降落，則額外獲得100分的獎勵。
只有總分達到200分或以上的回合被視爲解決方案。

根據以上規則和計分方式，在遊戲中，每個步驟都會獲得獎勵或減分，並且這些獎勵或扣分會影響整個回合的總分。通過合理的操作和策略，達到至少200分的總分便可視爲成功解決問題。

III. 研究方法與步驟

我們的架構圖如3所示，其中使用者可以透過PFRL框架設計一套符合馬可夫決策過程的強化學習環境，並使用PFRL建立agent物件與環境交互、學習，其中強化學習的環境採用Open AI gym[8]的LunarLander環境，如圖4，這是一個具有挑戰性的環境，透過強化學習算法學習控制環境中的主角順利且穩定地降落變能獲得分數，詳細規則與計分方式請見II. 背景知識與文獻探討，並且環境設計人員認爲累計分數達到200分便代表智能體可以從環境的反饋學習完整的遊戲規則。

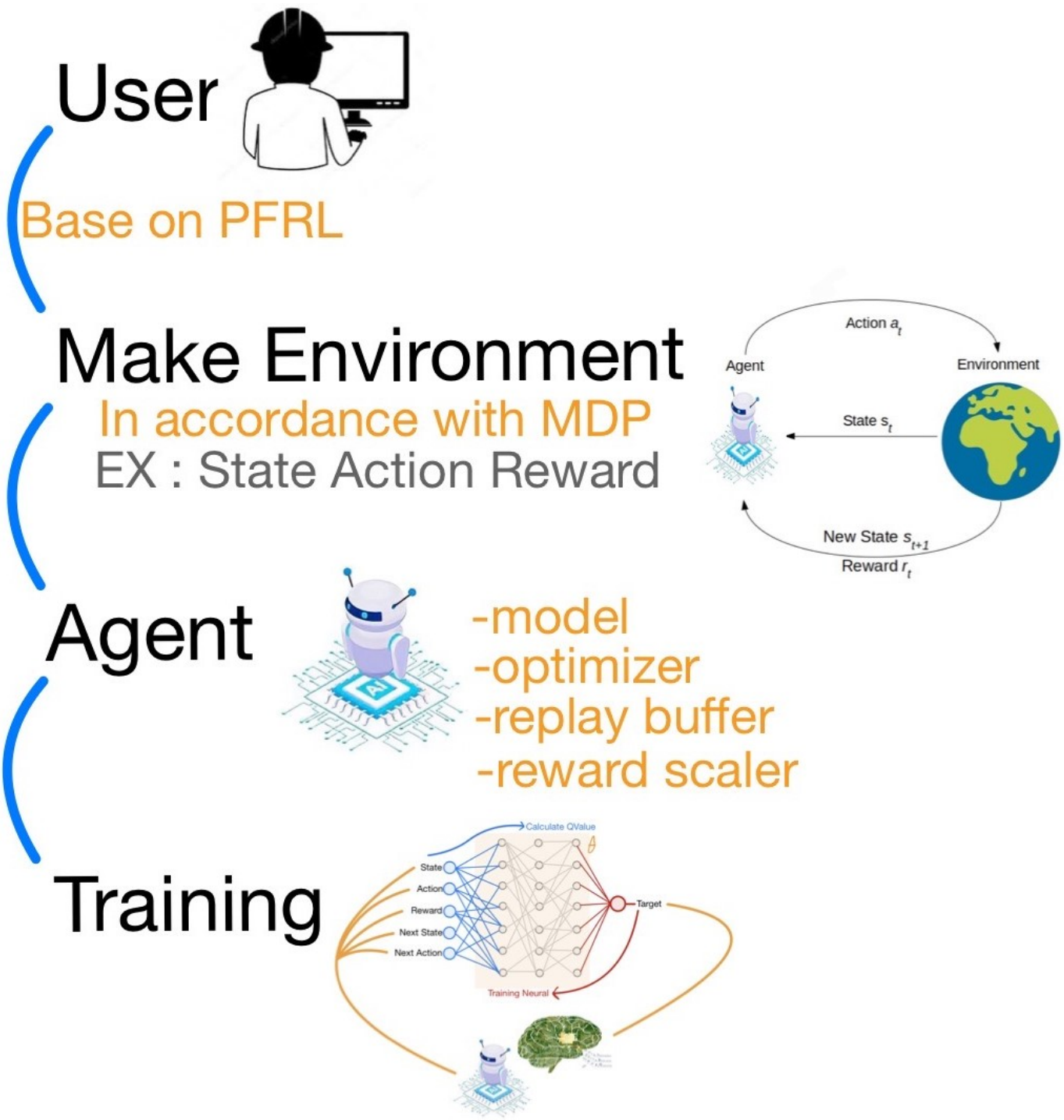
User
Base on PFRL
Make Environment
In accordance with MDP
EX : State Action Reward
Action a_t
Agent
Environment
State s_t
New State s_{t+1}
Reward r_t
Agent
-model
-optimizer
-replay buffer
-reward scaler
Training
Calculate QValue
State
Action
Reward
Next State
Next Action
Target
Training Neural

Fig. 3: 架構流程圖

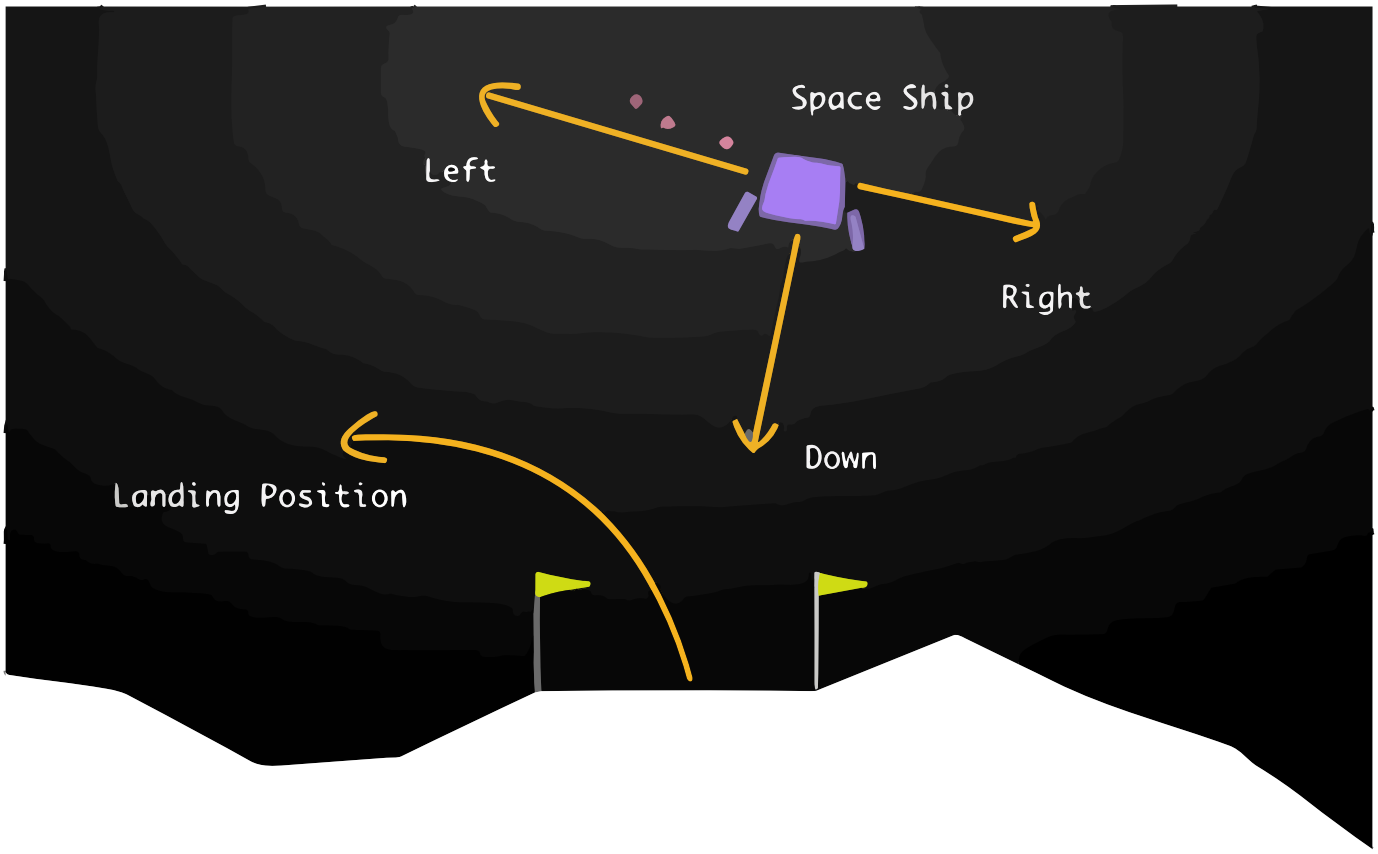

Fig. 4: LunarLander environment

A. 用*PFRL*演算法實現*LunerLander*

PFRL提供了完整的強化學習API，只需要套用演算法1的訓練架構便能開始進行智能體的訓練，其中只需要更換agent並輸入相對應的參數便能快速地實驗其他不同算法的效能。

B. *PFRL*演算法效能分析

爲了比較不同算法間的差異，我們將考量算法收斂所需的訓練次數、模型訓練完成後的平均分數以及針對智能體在完成單一回合所需的動作數量，我們稱它爲回合長度，觀察回合長度可以有效地判斷當前的學習狀態，例如:回合長度若一直過小可能是因爲智能體還沒學會控制"平衡"所以在回合開始後便墜毀，遊戲提早結束，回合長度較短。

1) 收斂條件設定*:* 強化學習中，由於演算法會不斷地強化分數較高的動作的Q值，進而引導智能體選擇期望值高的動作，然而這點也導致了Q值高估問題的出現，當訓練長度無止盡的拉長，便可能不斷地放大特定動作的Q值，最終造成模型損毀。所以需要合適第設定訓練結束的條件。公認的解決分數爲200，而我們爲了盡可能讓智能體獲得高分，以達到300分10次爲收斂條件，這個值界在過度訓練的邊緣，所以可以獲得更高的分數，但代價是一旦發生Q值高估問題，就只能重新訓練。另外，我們可以關注各算法達到收斂條件所需回合數，對於DQN而言，若參數設置不夠"精準"，一旦訓練回合過長去遲遲無法達到所設置的收斂條件便容易發生Q值高估問題；收斂所需的回合數也代表在相同參數設置下不同演算法的優劣程度，通常較高階的演算法，若參數設置妥當，可以以更少的訓練次數達到更高的分數。

Algorithm 1 PFRL實現DQN智能體並與環境互動

```
agent ← pfrl.agents.DQN(
    q_func,
    optimizer,
    replay_buffer,
    gamma,
    explorer,
    replay_start_size=500,
    update_interval=1,
    target_update_interval=10,
    phi=phi,
    gpu=gpu
)
for i ← 0 to n_episodes do
    obs ← env.reset()
    while True do
        env.render()
        action ← agent.act(obs)
        obs, reward, done, _ ← env.step(action)
        agent.observe(obs, reward, done, reset)
        if done or reset then
            break
        end if
    end while
end for
```

IV. 實驗與討論

A. 實驗環境

我們使用了各套件版本如下:

PFRL=0.3.0,
torch>=1.3.0,
gym>=0.9.7,
numpy>=1.10.4,

環境: LunarLander-v2及LunarLanderContinuous-v2[6]分別是離散動作空間與連續動作空間的環境，主要差別在於，連續的動作空間更考驗對智能體的訓練，在不改變環境(包括動作空間)的情況下，也就是參數配置

B. 實驗*1: DQN*-未設置中止條件

正如預期地，若不設置結束訓練條件最終便會發生Q值高估問題，導致模型損壞。如圖5所示:在約1800回合左右回合獎勵急遽下降，這就是典型的高估問題所導致，另外，可以注意到前面約500回合是在LunarLander環境下特有的變化，首先在訓練初期，智能體需要不斷地嘗試直到學習到"穩定飛行"，因爲緩慢且平穩的降落能獲得更高的分數，此時的回合長度變化應呈現從較低逐漸升高的趨勢，因爲未學會飛行的智能體往往會直接墜毀，回合長度短，這個階段通常只需較短的訓練次數就能學會；下一個階段則是學會穩定飛行後，學習降落在規定位置，這個階段是整個遊戲最核心的部分，因爲智能體只要較小的機率能剛好降落在目的地，而每一次落地失敗都將強化

智能體對"降落失敗導致扣分"的認知，若運氣不好，先前沒有降落成功的經驗，便會逐漸強化智能體"不要降落"的想法，最終將陷入區域最佳解，智能體只學會了穩定滯空，直到回合長度達到結束條件，所以這需要透過設置合適的超參數(例如:Discount factor、Learning rate或Exponential epsilon greedy decrement)避免；最後若能克服上一個階段，便能獲得約200分的成績，而最後階段便是學習，平穩、準確地落地提高額外的加分。另外爲了不同的目的訓練策略[7]也可分爲:

1. 穩健策略（降低月球登錄器的移動速度，緩慢靠近地面，防止墜毀）
2. 激進策略（只透過噴氣維保持平衡，接近地面時發動機全開，節省燃料）

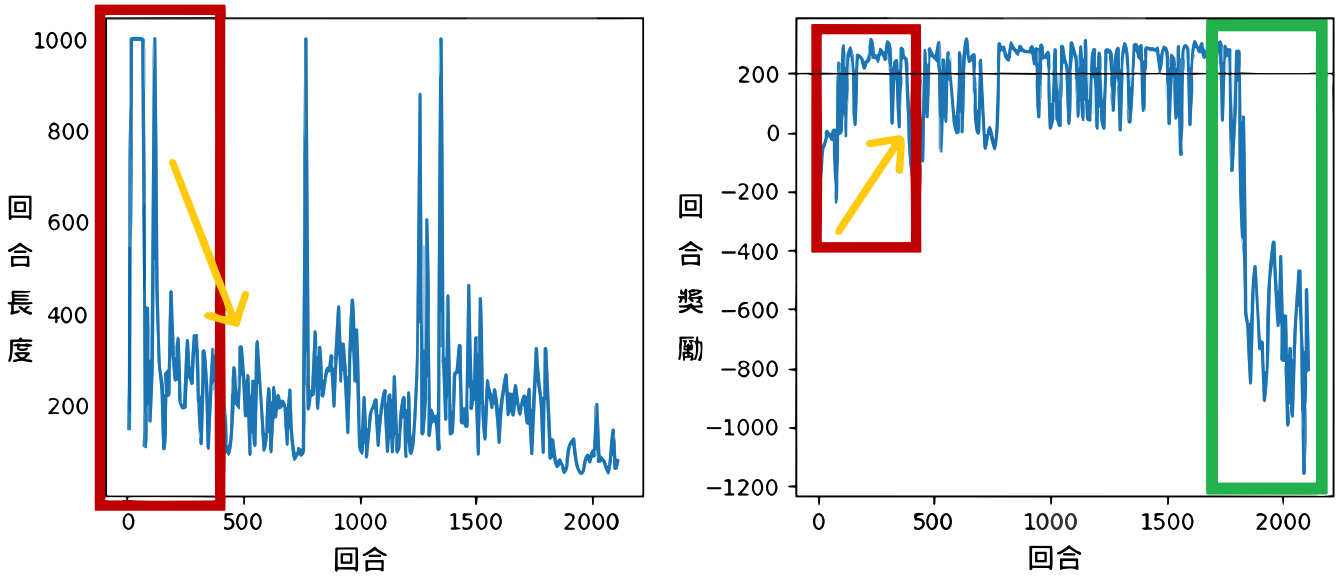

Fig. 5: DQN-未設置中止條件

C. 實驗2: *DQN與DDQN*

DQN與DDQN算法參數設置相同，如表II，經過訓練後，DQN智能體在經歷566回合的訓練後，平均可獲得265分，平均回合長度爲301。DDQN智能體在經歷532回合的訓練後，平均可獲得281分，平均回合長度爲216，如圖6、7，其中，由於 DDQN 引入了兩個獨立的神經網路(目標網路和決策網路)，使用目標網路來評估下一個行動的 Q 值，提高學習穩定性，目標網路的參數相對穩定，使用它來評估 Q 值可以減少訓練過程中的變動性，從而提高學習的穩定性。這有助於更快地收斂到更好的策略，這點也可以從獲得的獎勵看出。

TABLE II: DQN DDQN Hyperparameter

Parameter	Value
Memory size	1,000,000
Memory warmup size	500
Batch size	32
Learning rate	1e-3
Discount factor	0.99
Exponential epsilon greedy decrement	0.996
Start epsilon	0.99
End epsilon	0.01

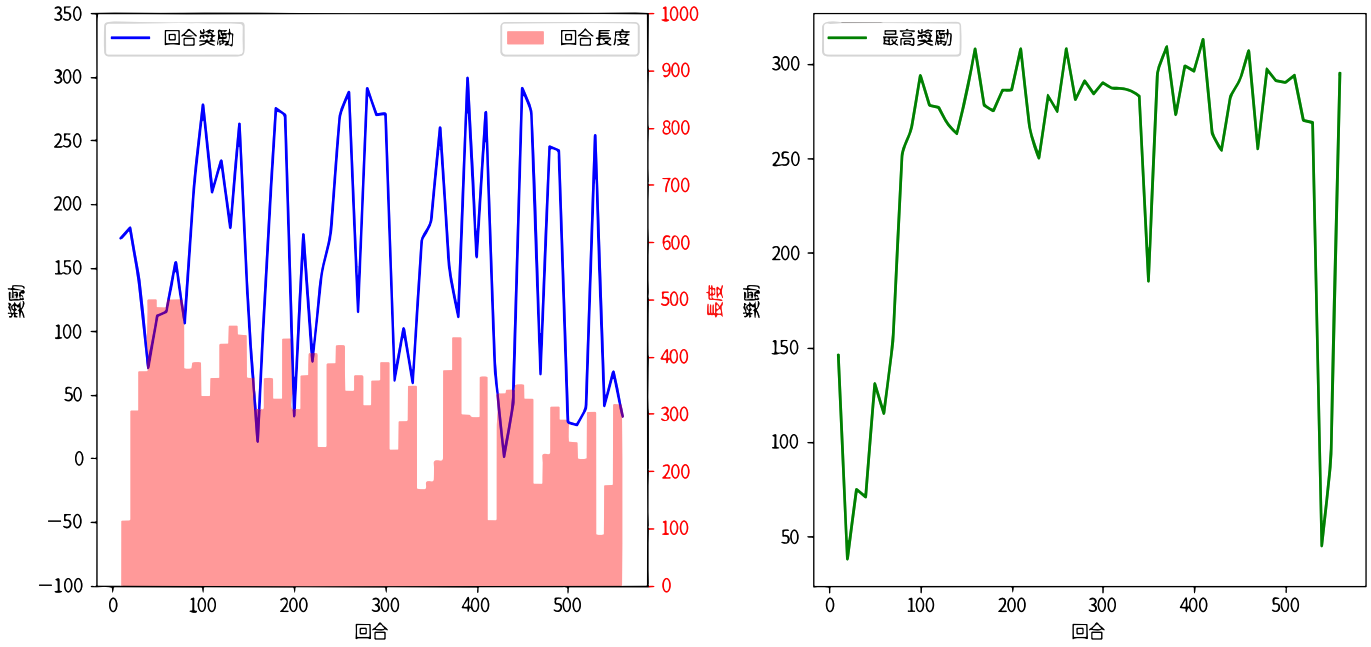

Fig. 6: DQN-agent 解決 LunarLander-v2

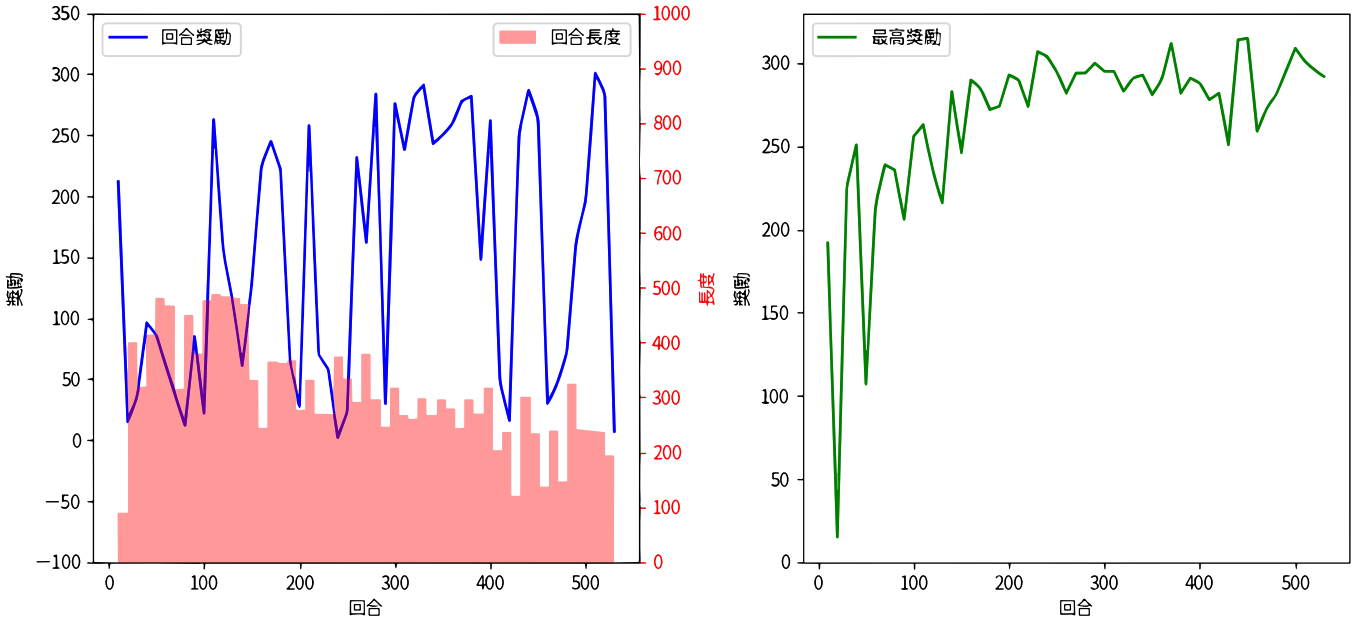

Fig. 7: DDQN-agent 解決 LunarLander-v2

D. 實驗3: DDPG

我們使用DDPG演算法並搭配更進階的LunarLanderContinuous-v2環境，再動作空間上爲連續的，也就是說更考驗了自能體對於不同狀態下選擇合適動作的難度，因爲相較於離散動作空間只需要選擇4種不同的動作後，便能清楚地得知最佳動作，而連續動作空間則備有無數種動作選擇，所以爲了學習最佳的決策往往更具以難度，因爲這容易導致陷入局部最佳解。

實驗結果證明，我們所選擇的參數組合能夠再連續動作空間的環境上獲得不錯的成績，具體來說，經過974回合訓練後智能體能只選擇262次動作，獲得269分的成績。如圖8。

TABLE III: DDPG Hyperparameter

Parameter	Value
Memory size	1,000,000
Memory warmup size	10000
Batch size	100
actor learning rate	2e-4
critic learning rate	3e-4
Discount factor	0.964
Exponential epsilon greedy decrement	0.996
Start epsilon	0.99
End epsilon	0.01
Soft update tau	5e-3

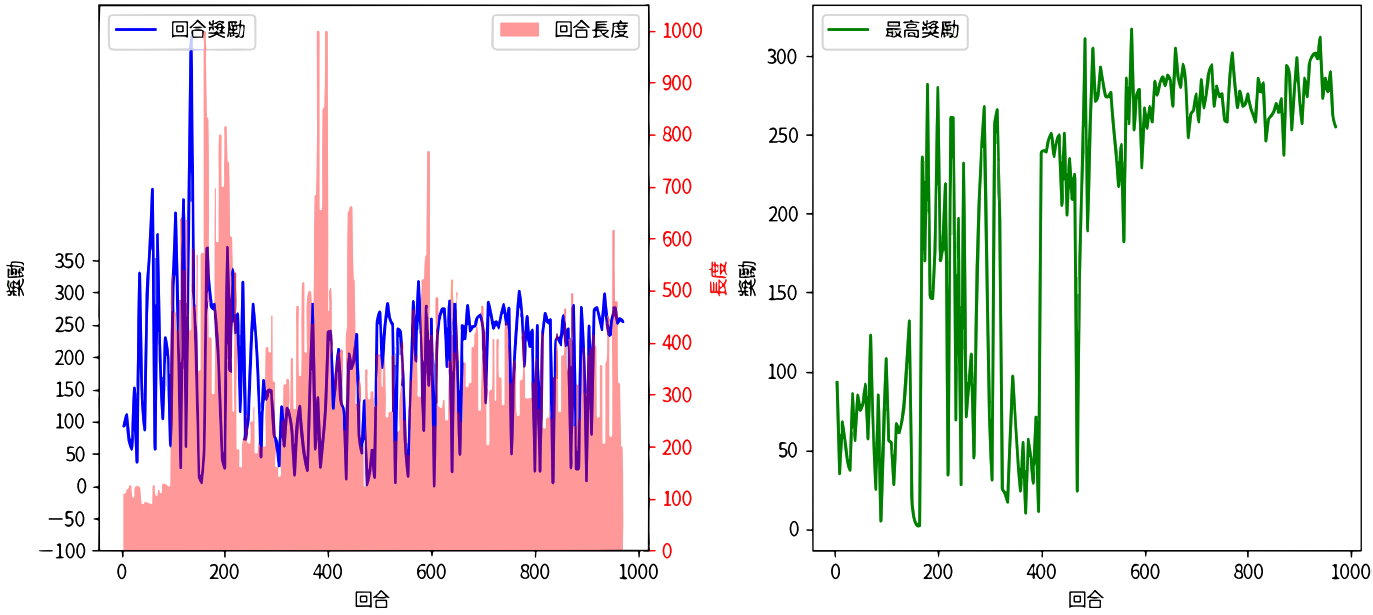

Fig. 8: DDPG-agent 解決 LunarLanderContinuous-v2

TABLE IV: 組員分工表

姓名	資料集	演算法	模型指標	其他
林廣哲	LunarLander-V2 LunarLanderContinuous-v2	1. DQN 2. DDQN 3. DDPG	分數 回合長度 收斂回合數	研究PFRL框架 實驗測量、結果繪製 教學影片
周名初		1. DDPG		繪製 架構流程圖 繪製 DRL運作圖
黃威綸		1. DQN		撰寫 研究背景 撰寫 研究動機
林維新		1. DQN		撰寫 研究目的 撰寫 研究問題

V. 結論

DRL是近年來人工智慧領域的研究焦點，但對初學者要求高且實作複雜。爲簡化開發，出現了多種DRL框架，其中PFRL是一個值得關注的框架。它易於開發，同時保有神經網路操作空間，對熟悉DRL的開發人員有幫助。然而，初學者可能覺得挑戰，因爲關於PFRL的教學資源較少。我們透過PFRL框架搭配OpenAI Gym函式庫的LunarLander環境，示範DQN、DDQN和DDPG三種常用的經典算法。在實驗中，我們的算法在得分上表現良好。這顯示我們的解決方案能夠在環境中有效學習並獲得較高的分數。對於DRL初學者，這提供了學習DRL完整程式架構的機會，並且我們也透過影片分享程式碼撰寫的過程及框架上需要注意的點，提供簡易且完整的PFRL框架入門教學，然而這些教學內容都建立在已經學習過強化學習算法Q-Learning的前提上，未來我們也希望可以進一步分享如何在PFRL框架上自定義環境與多智能體的環境探索。

REFERENCES

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. arXiv preprint arXiv:1312.5602.

[2] van Hasselt, H., Guez, A., Silver, D. (2015). Deep Reinforcement Learning with Double Q-learning. arXiv preprint arXiv:1509.06461.

[3] Bellemare, M. G., Dabney, W., Munos, R. (2017). A Distributional Perspective on Reinforcement Learning. arXiv preprint arXiv:1707.06887.

[4] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[5] Gym Documentation—LunarLander(https://www.gymlibrary.dev/environments/box2d/lunar_lander/)

[6] LunarLanderContinuous-v2(https://elegantrl.readthedocs.io/en/latest/tutorial/LunarLanderContinuous-v2.html)

[7] 化DelayDDPG，月球器，足机器人，只需半小的量、定代(https://zhuanlan.zhihu.com/p/72586697)

[8] OpenAI gym(https://github.com/openai/gym)

[9] PFRL(https://github.com/pfnet/pfrl)

[10] PARL(https://github.com/PaddlePaddle/PARL)

[11] karas(https://github.com/karasjs/karas)

[12] Pytorch(https://pytorch.org/)

[13] Gym Atari(https://www.gymlibrary.dev/environments/atari/index.html)