

輔仁大學資訊工程學系專題報告

利用演化多工增強式學習解決模擬控制機器 人問題

408470271 資工四甲 羅鈺婷 408470271@mail.fju.edu.tw
408261072 資工四甲 葉承翰 408261072@mail.fju.edu.tw
408290413 資工四甲 楊謹芳 408290413@mail.fju.edu.tw
408262648 資工四乙 劉懷萱 408262648@mail.fju.edu.tw

報告編號：CS111 年度-PR-A06

指導教授：廖容佐 博士

中華民國 111 年 12 月 27 日

摘要

近年來仿生機器人能支援越來越多種的工作，未來對機器人的需求也可能出現大幅成長，因此我們希望也能實現機器人的控制，由於我們沒有實際的機器人，所以我們透過 OpenAI Gym 的 Mujoco 模擬機器人模型作為實驗平台，而機器學習中的深度增強學習已在控制問題中有顯著的成果，因此以深度增強學習作為我們的實驗方向。本次專題以深度增強學習結合演化學習，也就是用 ERL 演算法和 CEM-RL 演算法為基礎，並加入演化多工的 SBO 演算法進行多任務間交互學習，而因為深度增強學習的訓練依然需要大量的時間，所以加入演化多工透過其他相關任務來學習以及增加訓練樣本，希望能夠達到提升訓練效能使訓練過程更有效率。我們將本專題所提出的方法和原本 ERL 演算法、CEM-RL 演算法以及在 ERL 演算法、CEM-RL 演算法所使用的深度增強學習演算法做比較，進行實驗結果的討論和分析。

目錄

摘要

目錄

圖目錄

表目錄

一、緒論	1
1.1 研究動機	1
1.2 專題議題	1
1.3 專題挑戰性	1
1.4 專題製作目標	1
1.5 專題貢獻	2
二、文獻探討	3
2.1 Deep Deterministic Policy Gradient(DDPG)	3
2.2 Twin Delayed Deep Deterministic Policy Gradient(TD3)	4
2.3 Soft-Actor Critic(SAC)	6
2.4 進化演算法(Evolutionary Algorithm)	7
2.5 演化式增強學習(Evolutionary Reinforcement Learning)	8
2.6 演化策略演算法(evolution strategies)和估計分佈演算法(Estimation of Distribution Algorithms)	9
2.7 交叉熵方法(Cross-Entropy Method)	9
2.8 CEM-RL	9
2.9 Symbiosis in Biocoenosis Optimization(SBO)	10
三、開發平台、程式語言與工具	12
3.1 OpenAI Gym	12
3.2 Multi-Joint dynamics with Contact (MuJoCo)	12
3.3 Anaconda	13
3.4 Python	13
四、實驗方法	14
4.1 ERL-SBO	14
4.2 CEM-RL-SBO	15
五、系統實作結果	16
5.1 實驗結果表	16
5.2 實驗結果圖	18
六、結論與未來展望	22
6.1 討論	22
6.2 未來展望	22
七、心得	23
八、參考資料	24
附錄	25
A. 安裝 Anaconda	25
B. 安裝 Openai gym	26
C. 安裝 MuJoCo	27
D. 安裝其他第三方套件模組	28

圖目錄

圖 1. DDPG 流程圖	4
圖 2. TD3 流程圖	5
圖 3. SAC 流程圖	6
圖 4. EA 流程圖	7
圖 5. ERL 流程圖	8
圖 6. CEM-RL 流程圖	10
圖 7. SBO 架構示意圖	10
圖 8. ERL-SBO 流程圖	14
圖 9. CEM-RL-SBO 流程圖	15
圖 10. HC (a)和 AT (b)	18
圖 11. HC (a)和 SW (b)	19
圖 12. HC (a)和 WK (b)	19
圖 13. SW (a)和 AT (b)	20
圖 14. SW (a)和 WK (b)	20
圖 15. WK (a)和 AT (b)	21
圖 16. 從官網下載 Anaconda 安裝包	25
圖 17. 建立 python 虛擬環境	25
圖 18. 激活虛擬環境	26
圖 19. 安裝 openai gym	26
圖 20. 下載 MuJoCo 安裝包	27
圖 21. 安裝 MuJoCo	27
圖 22. 測試是否安裝成功之程式碼	28
圖 23. 安裝 Tensorboard 套件模組	28

表目錄

表 1. SBO 共生關係	11
表 2. 演算法比較表之一	16
表 3. 演算法比較表之二	17
表 4. 演算法比較表之三	17
表 5. 演算法比較表之四	18

一、緒論

隨著人工智慧的應用越來越多元，我們發現近年來機器學習的應用領域有逐漸增加的趨勢，以及仿生機器人對於生活、工作方面的實用性，基於以上因素我們想要藉由 OpenAI Gym 的 MuJoCo 平台來測試和比較演算法對於提升模擬機器人的功效，本章節會依序說明此次專題的研究動機、專題議題、專題挑戰性、專題製作目標以及專題貢獻。

1.1 研究動機

近年來仿生機器人應用的領域越來越廣，我們看到了仿生機器人已經應用於負載重物、進行救災救援活動上，隨著對機器人的需求逐漸提升，我們也期望使用機器學習來實現機器人的控制。而機器學習(Machine Learning)可大致分為監督式學習(Supervised Learning)、非監督式學習(Unsupervised Learning)以及增強式學習(Reinforcement Learning)，因為深度增強學習已經在控制問題的領域有顯著的成果，因此使用深度增強式學習作為我們的實驗方向。

1.2 專題議題

在控制問題的領域上已經有許多深度增強學習的演算法問世，且都能達到不錯的成果，但深度增強學習的訓練過程需要花費龐大的時間和計算、對於稀疏獎勵的學習仍然是個議題。因此我們希望從現有的演算法中加入演化多工，透過任務間的學習達到增加訓練樣本、加快訓練過程的目的。

1.3 專題挑戰性

我們這組在開始做專題前，沒有學過任何機器學習相關知識也並不熟悉 Python，於是從深度增強學習演算法的相關論文開始，配合各種開源資源學習和請教指導教授，學習深度增強學習相關背景知識。而後我們為了在演化運算結合深度增強學習的基礎上，加入演化多工實現多任務學習，首要需完成多項任務同時訓練方式，但網路上大多資源都只有提到多代理的方式，所以花了很多時間不斷地嘗試修改程式碼，直到可以實現多項任務平行訓練。再來我們研讀了演化多工 SBO 演算法的論文，理解兩個任務交互學習的關係，並將其概念實踐於程式碼之上，最後解決不同維度的環境同時訓練、交換不同維度的個體問題。

1.4 專題製作目標

我們的目標是，在了解基本的深度增強學習演算法以及我們參考之論文所提供的 ERL 原始碼、CEM-RL 原始碼之後，將 SBO 演算法的演化多工概念融入其中，通過實驗，進行數據的比較與分析，找到更有效的訓練方式。

1.5 專題貢獻

本專題會利用一些現有的深度增強學習演算法、演化式深度增強學習和本專題提出之方法放在一些 OpenAI Gym 及 MuJoCo 平台之模擬機器人模型做比較，分析數據以及探討訓練不佳之因素，找出對於該任務中最有效之方法，以及對本專題所提出的方法之實驗數據結果做討論。

二、文獻探討

2.1 Deep Deterministic Policy Gradient(DDPG)

Deep Deterministic Policy Gradient(DDPG)演算法融合了策略評價網路(Actor-Critic)和 DQN(Deep Q-Learning)的經驗回放(experience replay)和目標網路(target network)的概念，最大的優點就是克服 DQN 演算法只能用於處理離散的動作空間，進而解決連續動作空間的問題。

在 DQN 中，曾使用到時序差分學習(Temporal Difference Learning)的概念，它利用代理(agent)觀測當前的狀態(state) S_t 並執行動作(action) A_t ，然後環境會給出新的狀態 S_{t+1} 並返回獎勵(reward) R_t 。TD target 的作法是把真正得到的獎勵 R_t 加上在 $t+1$ 時刻的預測乘上折扣因子(discount factor)，它會比在 t 時刻的預測更加精準。因此在 DDPG 中更新評價網路(Critic network)也用此法。在計算評分的時候，會希望評價 q_t 越接近 TD target 越好，換言之 TD Learning 的目的，就是讓代表 q_t 和 TD target 差距的 TD error，越小越好。

但 TD Learning 仍然有其缺陷，像是每用完一組 transition(s_t, a_t, r_t, s_{t+1})就將之丟棄，長此以往造成資料的浪費，還有按照順序使用每一條 transition 會因為前後資訊關聯性強，使訓練效果變差。為了解決這項缺陷，DDPG 加入一個回收緩衝(replay buffer)，用以 experience replay，它會把每個最新的 transition 存到 replay buffer 中，若 replay buffer 存滿了，每加入一條新的 transition 就會刪除一條舊的 transition，有了這個 replay buffer 除了可以將經驗重複利用，還能將順序打散，消除前後兩組 transition 的相關性。

關於 DDPG 中更新 Critic network 作法如同面提到的 TD Error。將輸入的動作和狀態進行評分，當作 target network 的輸出，並傳遞給 Critic network，讓他去計算 target value 接著再用 target value 減去 value evaluation 算出差距，求得 loss function 最小化的結果，以此提升 Critic network。

關於 DDPG 中更新策略網路(Actor network)的作法是當它接收到 Critic network 對於某一個動作可以帶來更好的評分時，Actor network 會朝著這個動作的方向去做梯度下降(Gradient descent)，以此對參數做更新。

關於 DDPG 中使用 target network 的作法是加上兩個跟 Actor 和 Critic 擁有一樣結構但參數不同的網路，因為在訓練 Critic network 的時候發現，如果一直使用同一個神經網路做預測，那麼高估評分或低估評分的誤差就會一直存在。所以 DDPG 把原本要用 Actor 和 Critic network 計算 $t+1$ 時刻評分的工作改由 target actor network 和 target critic network 來做。在更新 Target network 的時候，會先設定一個超參數，它的值在零與一之間，然後把 Critic network 的參數和 target critic network 的參數做加權平均，得到的值就可以當成新的 target critic network。而 target actor network 也一樣，把 Actor network 的參數和 target actor network 的參數做加權平均，得到的值就可以當成新的 target actor network。

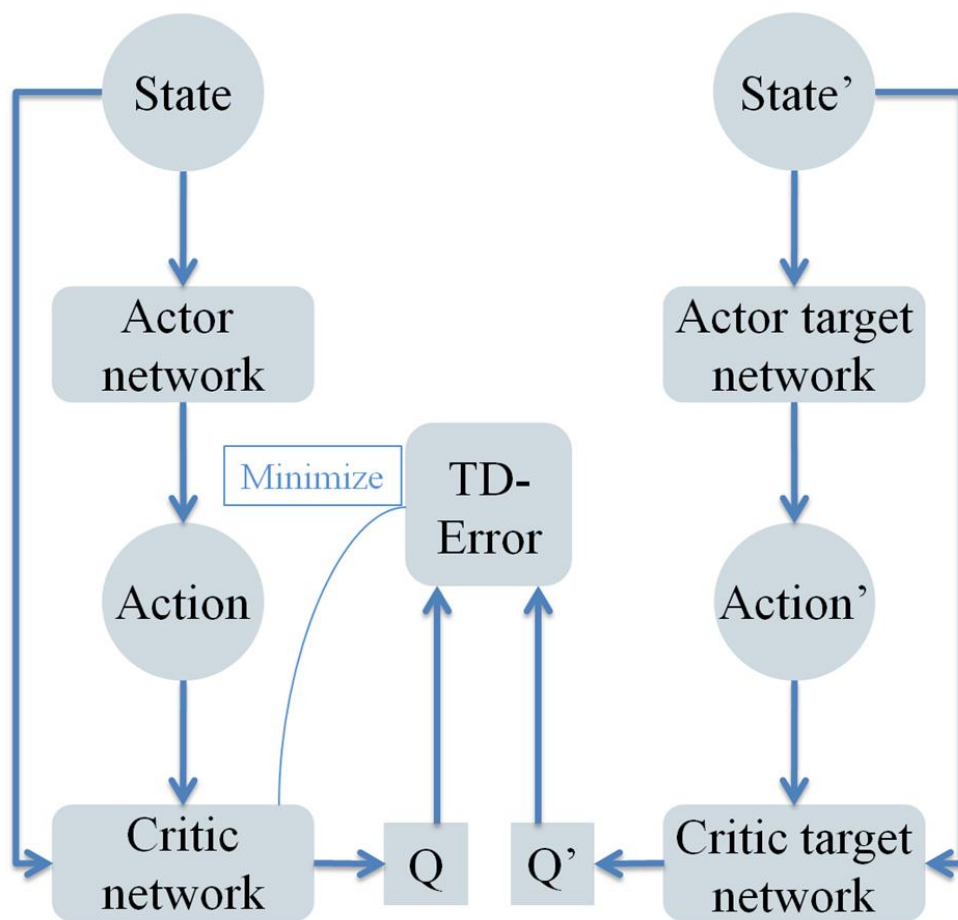


圖 1. DDPG 流程圖

2.2 Twin Delayed Deep Deterministic Policy Gradient(TD3)

TD3 是基於 DDPG 做優化，DDPG 源於 DQN，是解決連續控制問題的一種方法，然而 DDPG 也繼承了 DQN 會造成過度估計的問題。所以在 TD3 中引用 Double Q-learning 的思想，將選取動作(action)和估計分數(value)分別在預測網路(predict network)和目標網路(target network)上計算，有效改善了 DDPG 中預測 Q-Value 估計值過高的問題。

TD3 沿用了許多 DDPG 的做法，其中包含，更新評價網路(Critic network)的方式，使用時序差分學習(Temporal Difference Learning)，先透過觀測值算出 TD target 和 TD evaluation，相減並算出 TD Error。也使用 replay buffer 儲存經驗，而更新策略網路(Actor network)的作法是當它接收到 Critic network 對於某一個動作可以帶來更好的評分時，Actor network 會朝著這個動作的方向去做梯度下降(Gradient descent)，以此對參數做更新。TD3 中使用的 target network 的作法是加上兩個跟 Actor 和 Critic 擁有一樣結構但參數不同的網路，更新方式如同 DDPG。

TD3 主要新增了三項技術，第一，目標策略平滑(Target Policy Smoothing)，做法是在原本的 target network 的動作加上擾動(noise)，並且把 noise 的

範圍限制在一個區間內 $(-c, c)$ ，如此讓在下一個狀態的動作上加擾動的方法，可以有效避免極端值影響，讓評價網路的評估更準確。第二，剪裁的 Double Q-learning(Clipped Double-Q Learning)，在初始化時，賦予 Critic network 和 target critic network 不同的參數，每次更新的時候會得到不同的值(Q)，通過選取其中較小的值，作為更新的目標(target Q Value)。第三，延遲策略網路更新(Delayed Policy Updates)，不同於 DDPG，TD3 減緩了 Actor network 和 target network 的更新頻率，等 Critic network 提供的預測值比較準確後，再更新 Actor network，讓訓練更穩定，論文中是建議每更新兩次 Critic network 更新一次 Actor network。

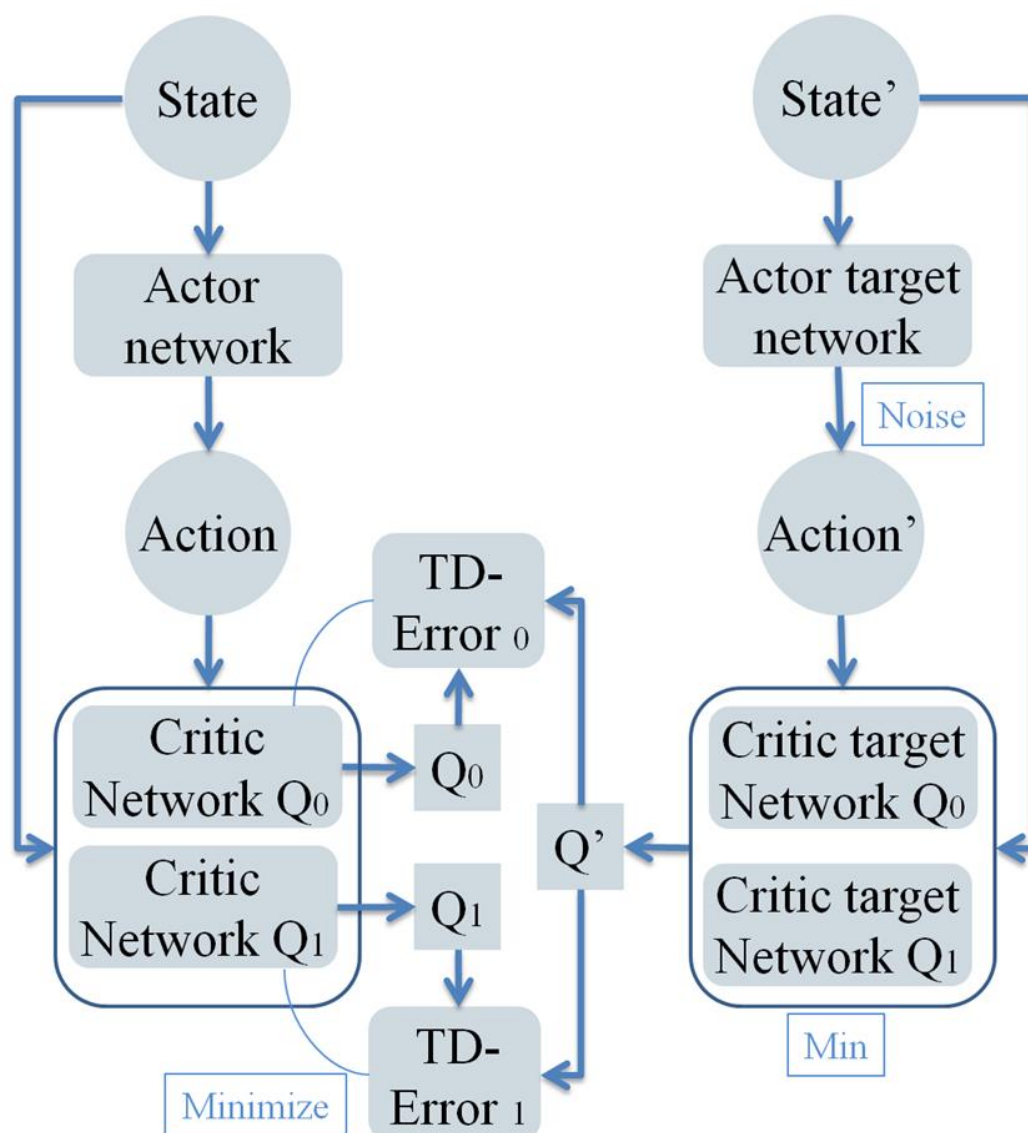


圖 2. TD3 流程圖

2.3 Soft-Actor Critic(SAC)

Soft Actor-Critic(SAC)演算法也是前面提到的 DDPG 延伸而來，但 DDPG 屬於 off policy 演算法，訓練的是一種確定性策略(deterministic policy)，也就是每個狀態都只考慮最好的一個動作稍微差的就不要了。而 SAC 和 DDPG 相比，使用的是隨機策略(stochastic policy)，隨機策略在機器人控制上通常是更好的做法。因為讓機器人完成一項任務，往往有許多條路徑去實現，而不會只有唯一的做法。因此，SAC 演算法是在每一個狀態都輸出每一種動作的機率，在多個 action 都是最佳解並且機率一樣大的時候，就可以從這些 action 中隨機選擇一個輸出。由此可見，SAC 中使用最大熵(maximum entropy)的觀念，盡量不浪費任何一個有用的 action，相比 DDPG 用的確定性策略具有一定的優勢，因為能夠探索更多的可能性。

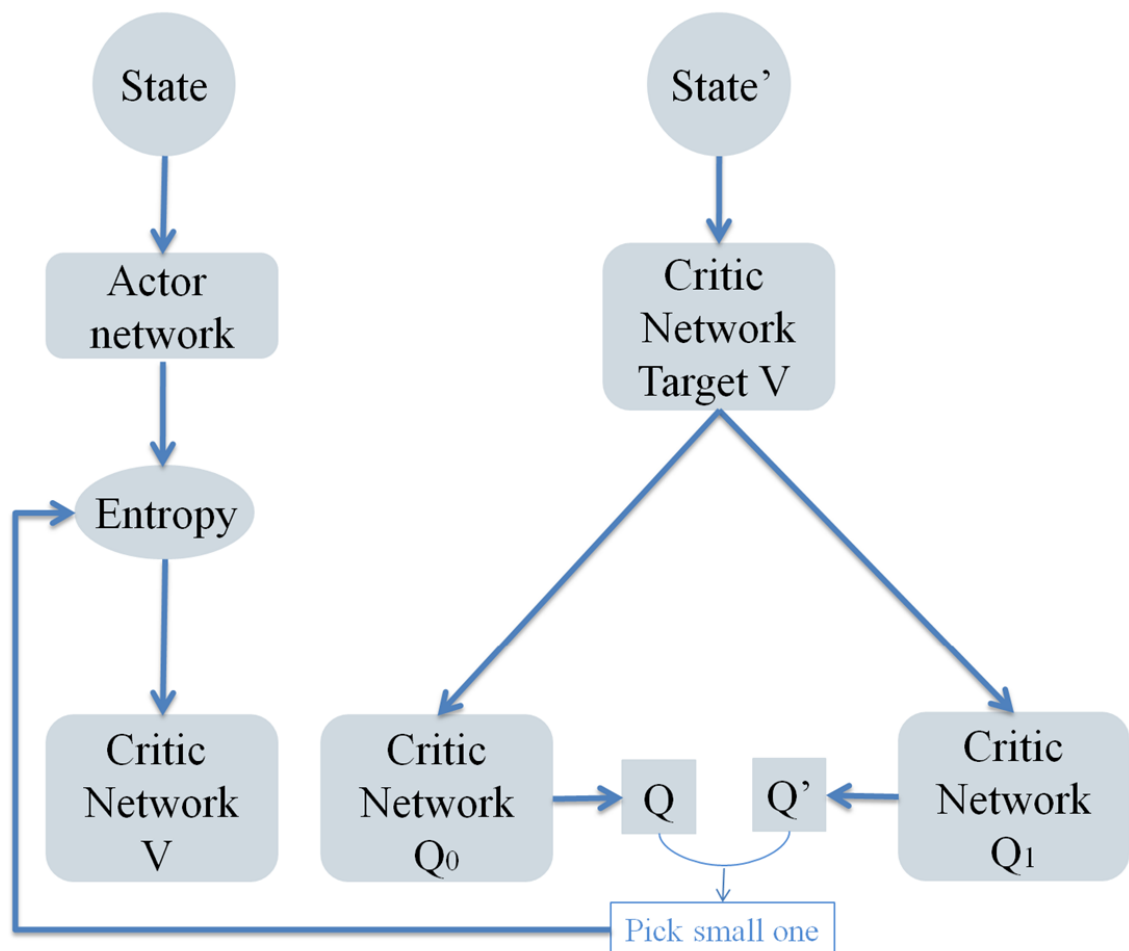


圖 3. SAC 流程圖

2.4 進化演算法(Evolutionary Algorithm)

進化演算法(Eiben & Smith, Introduction to evolutionary computing 2016)概念類似於達爾文的進化論，在找尋最佳解的過程，有如在一群隨機選擇的生物群中，對他們進行健康檢查，選擇最健康的生物的基因，使之進行繁殖，並隨著對環境的適應程度進行評估，淘汰最不合適在此環境生存的生物，周而復始。進化演算法，主要有四項運作，包括選擇(selection)，繁殖(new solution generation)，突變(mutation)和重組(solution alteration or recombination)。首先在一個空間中(search space)進行採樣，樣本的集合當作母體(population)，然後透過一個評價機制(fitness function)評價母體中每一個個體的適應度，選出合適的樣本，並進行變異(mutation)和重組(Recombination)，增加個體差異性，再透過評分高者，給予較高被選擇機會的選擇機制(selection)，成為子代。

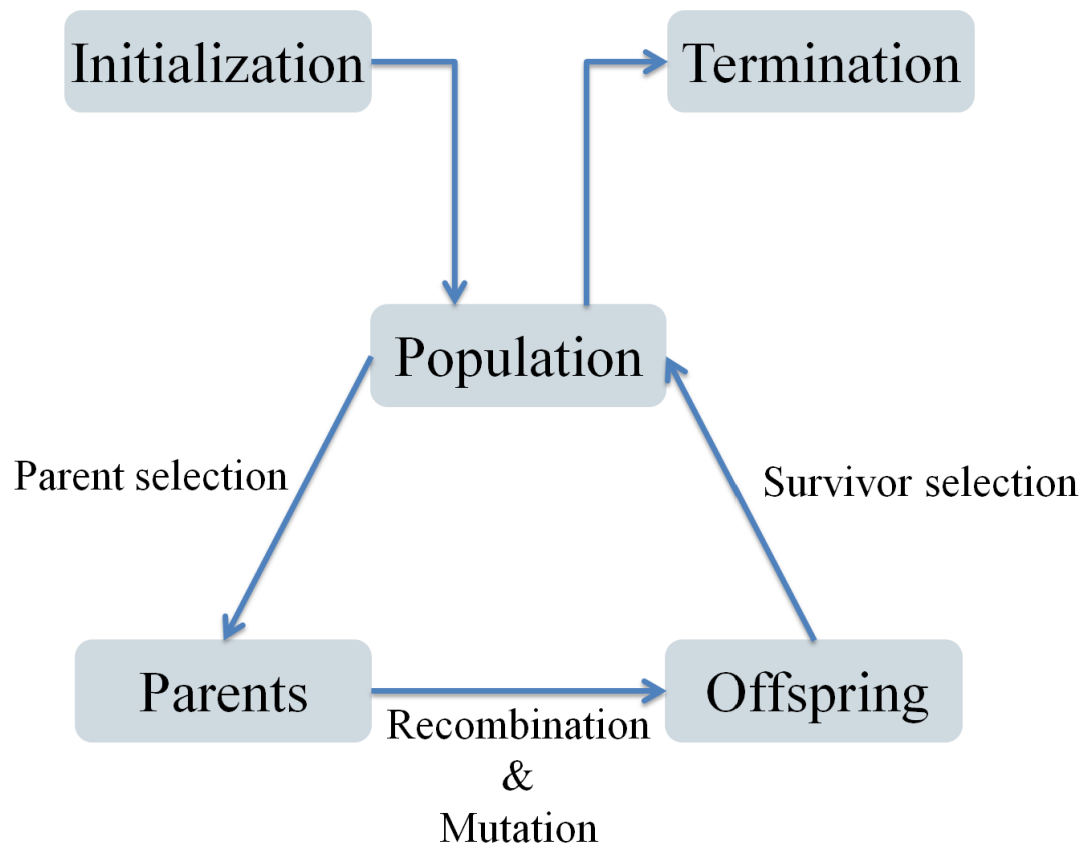


圖 4. EA 流程圖

2.5 演化式增強學習(Evolutionary Reinforcement Learning)

深度強化學習(DRL)演算法已經用於很多種控制任務。但是，這些方法依然會出現三個缺點：第一，具有稀疏獎勵的時間信用分配的問題，第二，缺乏有效探索，第三，對超參數非常敏感的收斂特性。整體而言，這些缺點，導致演算法對許多問題的適用度被大幅降低。而進化演算法(EA)是一個由達爾文的進化論啟發的技術，可以有效克服這三項缺點。但是，EA 通常很難解決需要最佳化許多參數的問題，因為樣本複雜度太高。於是出現了演化式增強學習(ERL)，它是一種混合 DRL 和 EA 的演算法，它利用 EA 的種群提供多樣化的數據來訓練 RL agent，並定時將 RL agent 放入 EA 的種群(population)中，讓 DRL 演算法的梯度資訊得以加入。ERL 繼承了 EA 具有時間信用分配的能力，還有多種策略能有效探索也更加具有穩定性，並且也繼承了 DRL 利用梯度提高學習效率的能力。在許多實驗中發現，ERL 明顯比 DRL 和 EA 訓練效果更好(而本專題 DRL 的部分主要是用 DDPG 和 SAC 演算法)。

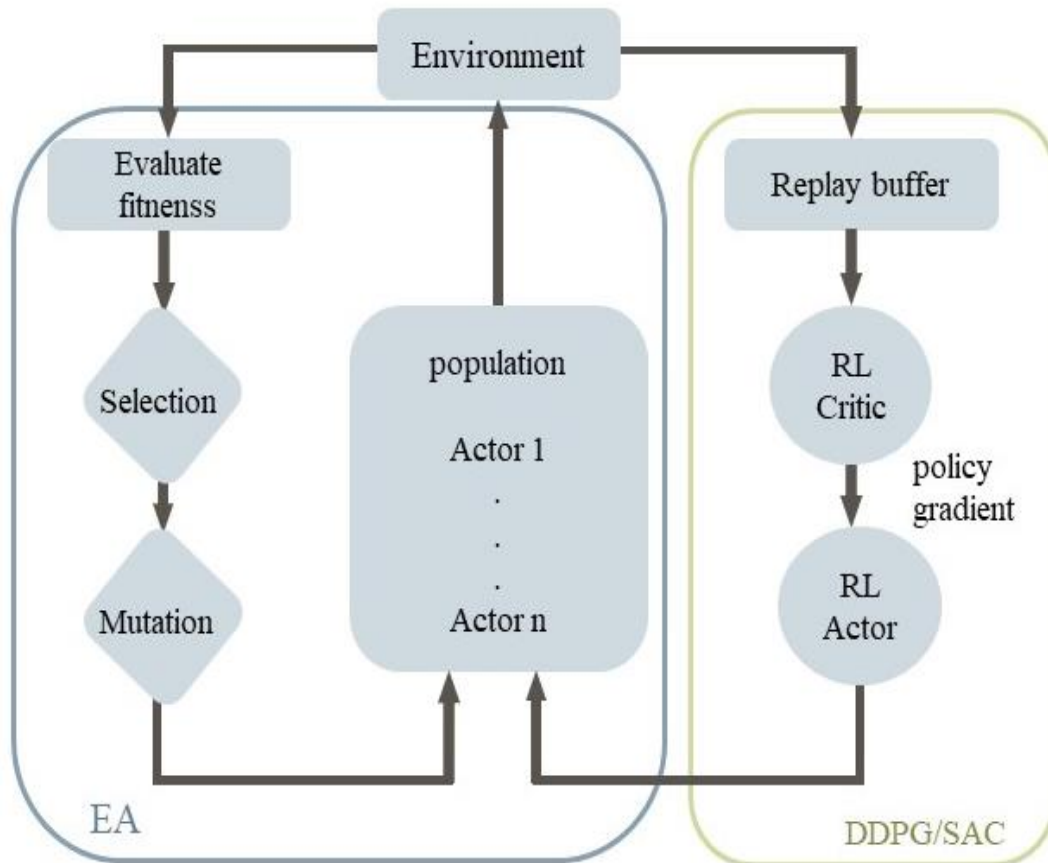


圖 5. ERL 流程圖

2.6 演化策略演算法(evolution strategies)和估計分佈演算法(Estimation of Distribution Algorithms)

演化策略演算法(ES)是進化演算法(EA)的延伸，主要不一樣的是，在每一個世代中，根據先前的資料計算出最優良的子代，唯一的最優良子代會被保留下來，並且再將其通過添加 Gaussian noise 來得到下一次的樣本。

在演化策略演算法中估計分佈演算法(EDA)是將 population 使用共變異數矩陣分佈(Larranaga&Lozano, Estimation of distribution algorithms: A new tool for evolutionary computation, 2001)這個共變異數矩陣定義了一個多變數的高斯函數而且也根據它定義了下一次迭代的樣本，在每一次的迭代由共變異數矩陣分佈的大範圍會逐漸縮小為局部最優解的較小範圍。

2.7 交叉熵方法(Cross-Entropy Method)

交叉熵方法(CEM) 是估計分佈演算法的一種，主要用來優化重要性採樣，交叉熵方法目標是最小化隨機得到的數據分佈與數據實際分佈與優良子代的距離，在每一次迭代的子代中其中優良的子代會被當作成精英，通常數量會取 population 的一半，精英會被用來計算下一代子代的平均以及變異數，而下一代會先被添加一些額外的變異數用來防止其過早收斂，再用精英的平均與變異數產出新一代的子代。

2.8 CEM-RL

CEM-RL 是一種結合了進化策略和以梯度方法為基礎的策略搜尋。策略搜尋會最大化尚未搜尋過的部分以得到最大的 reward。而進化策略是利用類似於生物演化的過程，在此過程中，會將好的部分留下，剩下的部分則會透過交叉操作(crossover)和突變(mutation)重新產生，再去跟之前好的部分比較，進而留下最好的部分。而梯度方法可以讓進化策略(Evolution Strategy, ES)更加穩定且帶來更好的樣本效率(sample efficiency)。

如下圖，CEM-RL 主要分為三個部分，分別為 CEM、DDPG/TD3 和 Evaluation。

一開始會先初始化 random actor，每次迭代會把 population 分為兩半，其中一半會先送到 Evaluation 計算 fitness，而另一半則會被送到 DDPG/TD3 的部分去更新 Critic，再去計算 fitness。最終，我們會取 fitness 較高的一半當作 parent 透過將高斯噪聲(Gaussian function)加到目前最好的 Actor。最後，在進化策略(ES)中，會使用機率分布估計演算法(estimation of distribution algorithm, EDA)中的斜方差矩陣 Σ (Larranaga & Lozano, 2001)定義高斯函數，再去產生新的 population，這樣就算完成一個迭代。

機率分布估計演算法(EDA)全名：estimation of distribution algorithm，是一種演化演算法。其特點在於，他會對整個空間蒐集樣本在利用統計學來預判目前最好的方法。和 GA 比較起來，因為 EDA 是搜尋整個空間，樣本會比較齊全，因此搜尋能力以及收斂的速度會比 GA 好。

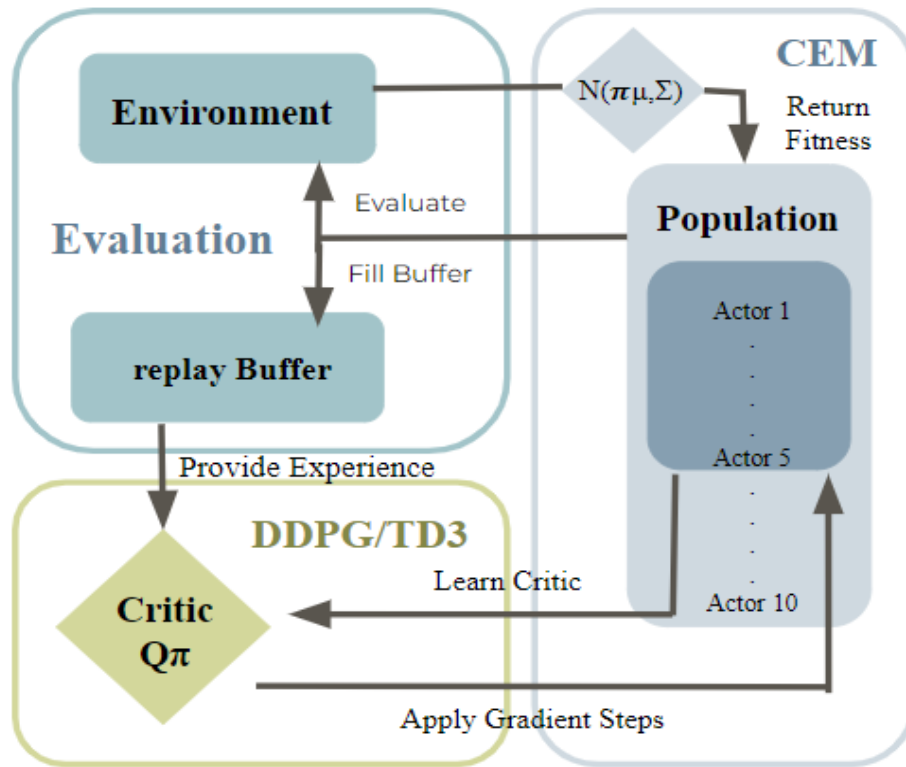


圖 6. CEM-RL 流程圖

2.9 Symbiosis in Biocoenosis Optimization(SBO)

SBO 演算法為演化多工的一種，SBO 有多個 EAs，每個 EA 的種群(population)稱為生物群落(Biocoenosis)，而生物群落之間的關係稱為共生關係(Symbiosis)，因此 SBO 就是透過生物群落間的共生關係來達到資訊傳遞。

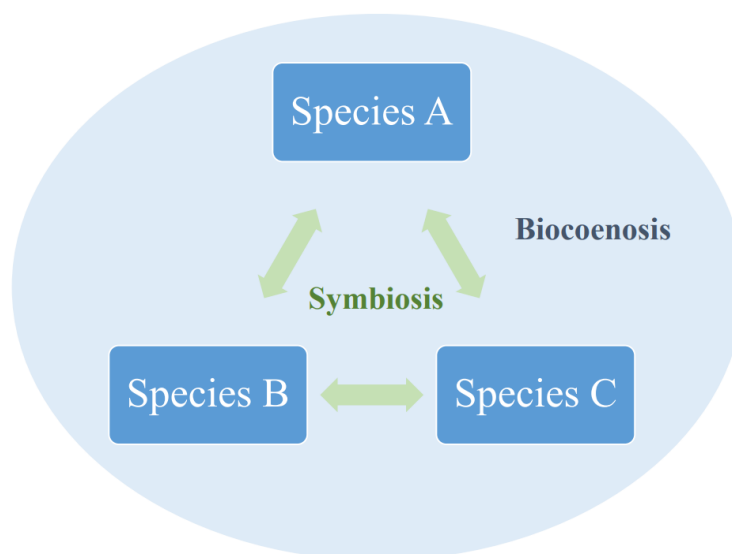


圖 7. SBO 架構示意圖

在 SBO 中有 m 個任務就有 m 個 EAs，每次的迭代每個 EA 都會做資訊的傳遞，而在 SBO 中傳遞資訊則是透過交換子代個體來達成。透過 A 物種對 B 物種有益(Beneficial)/有害(Harmful)/普通(Neutral)的關係，定義了兩個生物群落間的共生關係。

首先介紹 Beneficial、Harmful、Neutral 的定義，其中 N 為 population 數量， $\beta = 0.25$ ， $H = 0.5$

Beneficial：個體 c 適應值排名在任務 τ 中的前 βN 個，以 $c > \tau$ 表示

Harmful：個體 c 適應值排名在任務 τ 中的後 HN 個，以 $c < \tau$ 表示

Neutral：不屬於以上兩者，以 $c \approx \tau$ 表示

再來以 Beneficial、Harmful、Neutral 定義個體 c 對任務 A 與 B 兩個群落間的共生關係，共有六種情況

Mutualism (M)：A 與 B 互利， $c > A$ 和 $c > B$

Neutralism (N)：A 與 B 無關， $c \approx A$ 和 $c \approx B$

Competition (C)：A 與 B 互害， $c < A$ 和 $c < B$

Commensalism (O)：A 對 B 有益，B 對 A 普通， $c > A$ 和 $c \approx B$

Parasitism (P)：A 對 B 有益，B 對 A 有害， $c > A$ 和 $c < B$

Amensalism (A)：A 對 B 普通，B 對 A 有害， $c < A$ 和 $c \approx B$

表 1. SBO 共生關係

		任務 A		
		Beneficial	Neutral	Harmful
任務 B	Beneficial	M	O	P
	Neutral	O	N	A
	Harmful	P	A	C

而透過共生關係就能計算出轉移率(transfer rate)，而 transfer rate 就會影響到下一次迭代時交換的個體數目，交換子代後就會進行生存競爭，目的是將整體 population 推向好的解，因此這樣的流程就依照這樣繁衍子代、資訊傳遞、生存競爭繼續下去。

三、開發平台、程式語言與工具

3.1 OpenAI Gym

OpenAI gym 提供了許多模擬環境，主要可以分成以下五種類別，分別是 Atari、MuJoCo、Toy Text、Classic Control、Box2D，使用者可以透過這些模擬環境，來測試演算法的訓練成效，因此成為一個十分方便的介面，所以我們選用它來做測試。

由於我們期望實現仿生機器人的控制，所以選用 MuJoCo 類別中的模擬環境，MuJoCo 是一個物理模擬引擎，後面會詳細介紹。

3.2 Multi-Joint dynamics with Contact (MuJoCo)

我們日常生活中舉手投足間，每個不經意的動作，貌似稀鬆平常，但從物理世界的微觀角度而言，其實是十分精細的動作，比如我們的觸覺，能感知到溫度、濕度、光滑或粗糙、柔軟或堅硬等等。在觸覺的感測下，準確的觸碰或拿取我們想要的東西。而 MuJoCo 這個物理模擬引擎，就有如我們人類的感官功能，賦予機器人感測的能力，讓機器人更精確地模擬人類或動物的行為動作。

我們主要的實驗環境為 MuJoCo 類別下的 Ant-v2、HalfCheetah-v2、Swimmer-v2、Walker2d-v2，以下一一介紹。

- Ant-v2

樣式：是一個 3D 機器人，由一個可轉動的身體和四條腿組成。每條腿上有兩個連接點。

目標：通過訓練，學會在樞紐處施加合適的力道，控制四條腿穩定地向前移動。

- HalfCheetah-v2

樣式：是一個 2D 機器人，身體可以分成 9 個部份，由 8 個關節做連接，軀幹和頭部位置是固定的，只能施力在其他 6 個關節上，分別是大腿前後連接到軀幹的部分、小腿連接到大腿的部分和腳連接到小腿的部分。

目標：通過訓練，學會在關節上施加合適的力道，使獵豹盡可能快地向前跑，根據向前移動的距離分配正獎勵、向後移動分配負獎勵。

- Swimmer-v2

樣式：是一個 2D 機器人，由三個或更多段的軀幹，和軀幹數量減一個關節組成，一個關節正好連接兩個軀幹，使身體形成線性鏈。

目標：通過在關節上施力，並克服流體摩擦來儘快向前移動。

- Walker2d-v2

樣式：是一個 2D 機器人，由四個主體部分組成，頂部的軀幹，兩條大腿連接軀幹下方，兩條小腿連接大腿下方，兩隻腳連接到小腿。通過六個關節，連接身體部位並施加力矩來定向。

目標：協調兩組腳、小腿和大腿快且穩地向前移動。

3.3 Anaconda

Anaconda 是一款開源的 Python 和 R 語言發行版本，普遍用於處理大量資料以及計算科學，擁有相當強大的套件管理功能，提供許多 Python 常用科學計算套件，支援 Windows、macOS 和 Linux 作業系統，也提供 Jupyter Notebook、Spyder 等程式開發環境。

3.4 Python

Python 是一種物件導向程式語言，同時也是一種廣泛使用的直譯式、進階和通用的程式語言，通常被用來做數據分析、網頁開發應用、人工智慧應用和資料科學……等非常多的領域，因為其開源套件多的特性，使 Python 成為目前機器學習和深度學習最熱門的語言。

四、實驗方法

4.1 ERL-SBO

本專題提出之第一種實驗方法是以 Evolutionary Reinforcement Learning (ERL) 為基礎進行演化以及產生子代，和 symbiosis in biocoenosis optimization (SBO) 交換各任務中之子代個體這兩部分所組成。

ERL 是一種混合式的演算法，利用進化演算法(EA)中基於種群(population)的方法生成不同的訓練樣本，提供給 RL 代理做訓練，在定期地將透過深度增強學習(DRL)所訓練之 Actor 網路(RL Actor)放入 population 中，讓策略梯度訊息得以加入，這邊所用之 DRL 演算法為 DDPG 以及 SAC 演算法，而 ERL 擁有 EA 中根據適應度來選擇的步驟則可以淘汰適應度低的個體、留下高適應度個體。ERL-SBO 的流程會在一開始以隨機權重初始化 population 中的十個 Actor 網路，以及一個額外以 DRL 所訓練的 Actor 網路和 Critic 網路一起初始化，在每一代中，會計算 population 中的每個個體適應度，並取前兩高適應度的個體當作菁英保留下來，其中適應度是以 Actor 每回合和環境互動得到之 Reward 總和，而之後選擇機制會選擇 population 中部分之個體透過交叉操作(crossover)和突變(mutation)進行擾動，最後再加入一個 RL Actor，因此子代就會由保留下來的精英、擾動過的個體以及 RL Actor 所組成進入到演化多工進行多任務處理。

之後將兩個任務中透過 ERL 演算法產生的兩組子代進入演化多工之 SBO 演算法中，並在一定的機率下交換兩任務子代中的某些個體，再計算從別的任務中換來的新個體適應度，根據計算到的適應度決定換來的個體之表現進而更新共生關係，再依照新的共生關係就能計算每個任務之轉移率，且轉移率就會影響到下一代交換的個體數目，就這樣新的子代就會成為下一代的 population，而整體就以這樣的流程不斷重複地一直迭代下去。

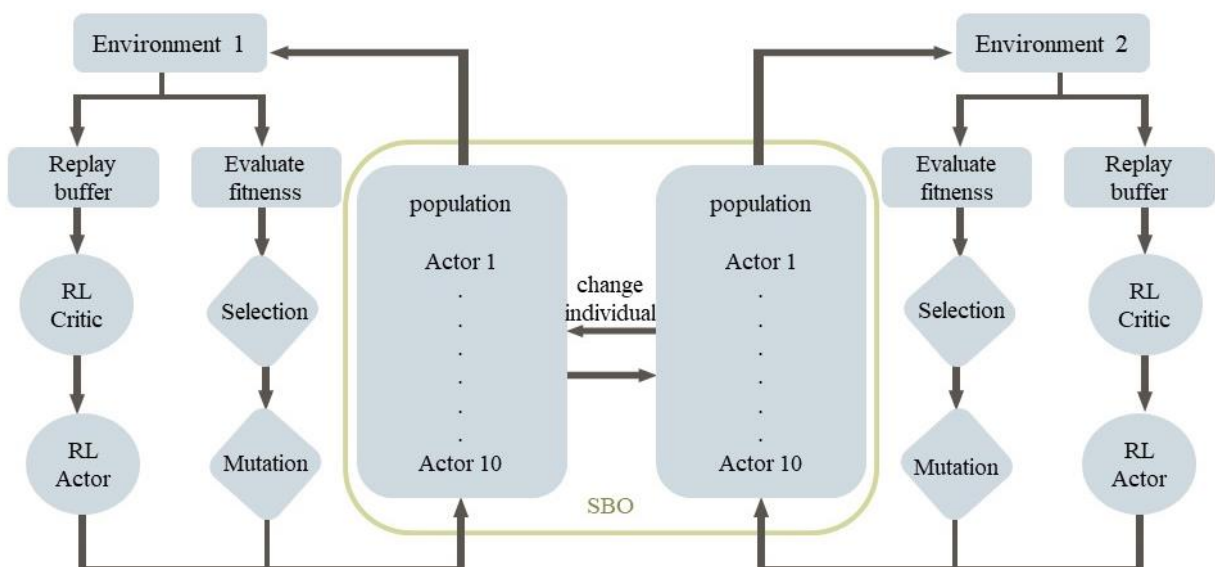


圖 8. ERL-SBO 流程圖

4.2 CEM-RL-SBO

本專題提出的第二種實驗方法是，是用 CEM-RL(Combining evolutionary and gradient-based methods for policy search)更新子代，再利用 SBO (Symbiosis in Biocoenosis Optimization)交換兩個環境的子代，以達到更好的表現。

CEM 一開始會先初始化 random actor，然後在每次迭代時會把種群(population)分為兩半，其中一半會先送到 Evaluation 計算個體適應度，而另一半則會被送到 DDPG/TD3 的部分用 Critic 更新後，再重新計算個體適應度。最終，我們會取個體適應度較高的一半當作父代，然後透過高斯分布再去產生新的種群。

而 SBO 會作用在評估完個體適應度之後，用來交換兩個任務中的子代，然後再計算各子代的個體適應度，藉此來評估交換來的個體表現，分為：好、普通及差。接著，再根據個體的表現更新這兩個任務的共生關係。

最後，透過先前得到的共生關係，重新評估著兩個任務的轉移率，減少或是增加交換的子代，所以這會影響到下一次迭代時交換個體的數目以更新 population，然後再重複 CEM-RL 原本的步驟，再次用 Critic 更新及評估個體適應度。

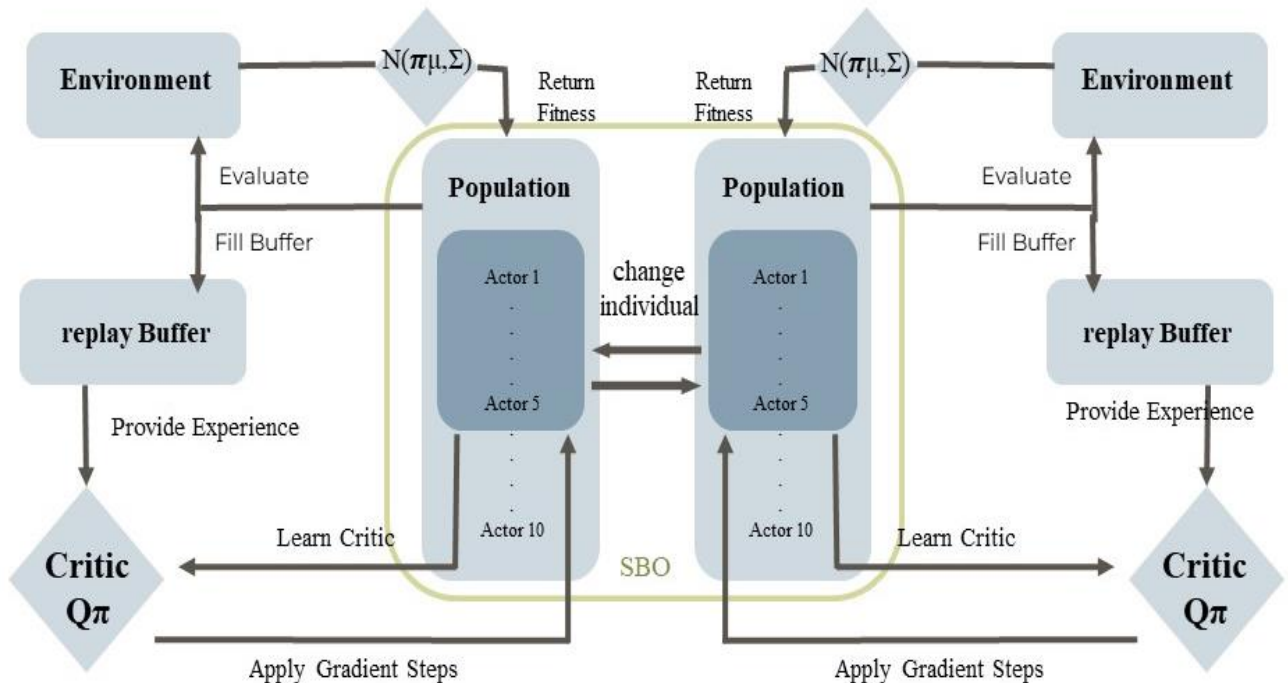


圖 9.CEM-RL-SBO 流程圖

五、系統實作結果

本專題以 OpenAI Gym 中 MuJoCo 的 Ant-v2、HalfCheetah-v2、Swimmer-v2、Walker2d-v2 共四個環境作為我們的測試平台，以下將這四個問題簡寫為 Ant-v2(AT)、HalfCheetah-v2(HC)、Swimmer-v2(SW)、Walker2d-v2(WK)，其中在演化多工的部分是以兩個環境之間做學習，因此在四組環境中選出兩個環境總共會有六個組合，分別是 HC 和 AT、HC 和 SW、HC 和 WK、SW 和 AT、SW 和 WK、AT 和 WK。

5.1 實驗結果表

以下列出各方法之實驗數據比較表，我們是以五個亂數種子進行實驗，每次實驗跑一百萬 steps，將每次實驗最後一筆數據用來計算出其平均數，其中的 P 值是以統計分析之 Wilcoxon rank sum test 方法計算而得的。

表 2. 演算法比較表之一

P	T		Mean			P value	
			DDPG	ERL	SBO	DDPG vs. SBO	ERL vs. SBO
P1	T1	HC	1586.87	5082.80	4981.27	0.004	0.345
	T2	AT	502.94	944.81	1009.48	0.075	0.111
P2	T1	HC	1586.87	5082.80	5213.46	0.004	0.21
	T2	SW	141.74	330.43	352.41	0.004	0.028
P3	T1	HC	1586.87	5082.80	4671.11	0.004	0.210
	T2	WK	890.46	1038.39	983.15	0.345	0.210
P4	T1	SW	141.74	330.43	349.71	0.004	0.075
	T2	AT	502.94	944.81	867.24	0.075	0.210
P5	T1	SW	141.74	330.43	345.64	0.003	0.155
	T2	WK	890.46	1038.39	990.60	0.500	0.421
P6	T1	AT	502.94	944.81	847.02	0.111	0.421
	T2	WK	890.46	1038.39	1019.31	0.274	0.421

在 DDPG 的部分，比較 ERL 和 SBO 的平均數，有 7 個輸，5 個贏。

表 3. 演算法比較表之二

P	T		Mean			P value	
			SAC	ERL	SBO	SAC vs. SBO	ERL vs. SBO
P1	T1	HC	4756.80	6417.00	5836.59	0.155	0.111
	T2	AT	4943.51	1118.94	1056.94	0.004	0.274
P2	T1	HC	4756.80	6417.00	6831.65	0.008	0.345
	T2	SW	34.19	240.48	269.96	0.004	0.345
P3	T1	HC	4756.80	6417.00	6328.04	0.028	0.500
	T2	WK	3968.27	1673.11	1969.35	0.004	0.500
P4	T1	SW	34.19	240.48	205.40	0.004	0.345
	T2	AT	4943.51	1118.94	1027.47	0.004	0.075
P5	T1	SW	34.19	240.48	185.63	0.004	0.155
	T2	WK	3968.27	1673.11	1180.08	0.004	0.155
P6	T1	AT	4943.51	1118.94	1148.01	0.004	0.274
	T2	WK	3968.27	1673.11	1014.22	0.006	0.104

在 SAC 的部分，比較 ERL 和 SBO 的平均數，有 8 個輸，4 個贏。

表 4. 演算法比較表之三

P	T		Mean			P value	
			TD3	CEM	SBO	TD3 vs. SBO	CEM vs. SBO
P1	T1	HC	7392.01	10774.14	11100.59	0.004	0.274
	T2	AT	3055.13	3706.62	4329.73	0.004	0.274
P2	T1	HC	7392.01	10774.14	11477.83	0.004	0.028
	T2	SW	70.21	92.25	99.15	0.048	0.274
P3	T1	HC	7392.01	10774.14	10279.28	0.048	0.274
	T2	WK	3145.67	3484.86	4416.01	0.048	0.274
P4	T1	SW	70.21	92.25	125.99	0.075	0.210
	T2	AT	3055.13	3706.62	4279.74	0.075	0.274
P5	T1	SW	70.21	92.25	89.91	0.111	0.500
	T2	WK	3145.67	3484.86	3422.20	0.345	0.345
P6	T1	AT	3055.13	3706.62	4916.75	0.008	0.028
	T2	WK	3145.67	3484.86	3737.13	0.210	0.210

在 TD3 的部分，比較 CEM 和 SBO 的平均數，有 3 個輸，有 9 個贏。

表 5. 演算法比較表之四

P	T		Mean			P value	
			DDPG	CEM	SBO	DDPG vs. SBO	CEM vs. SBO
P1	T1	HC	1586.87	9144.49	10706.80	0.003	0.075
	T2	AT	502.94	637.83	1121.97	0.421	0.500
P2	T1	HC	1586.87	9144.19	8821.24	0.003	0.345
	T2	SW	141.74	182.38	216.93	0.075	0.155
P3	T1	HC	1586.87	9144.49	9867.10	0.004	0.273
	T2	WK	890.46	1300.22	1627.44	0.016	0.210
P4	T1	SW	141.74	182.38	161.73	0.210	0.500
	T2	AT	502.94	637.83	839.19	0.154	0.421
P5	T1	SW	141.74	182.38	166.21	0.028	0.345
	T2	WK	890.46	1300.22	1508.94	0.075	0.274
P6	T1	AT	502.94	637.83	495.06	0.274	0.210
	T2	WK	890.46	1300.22	1647.84	0.278	0.155

在 DDPG 的部分，比較 CEM 和 SBO 的平均數，有 4 個輸，有 8 個贏。

5.2 實驗結果圖

(1) HC 和 AT 組合

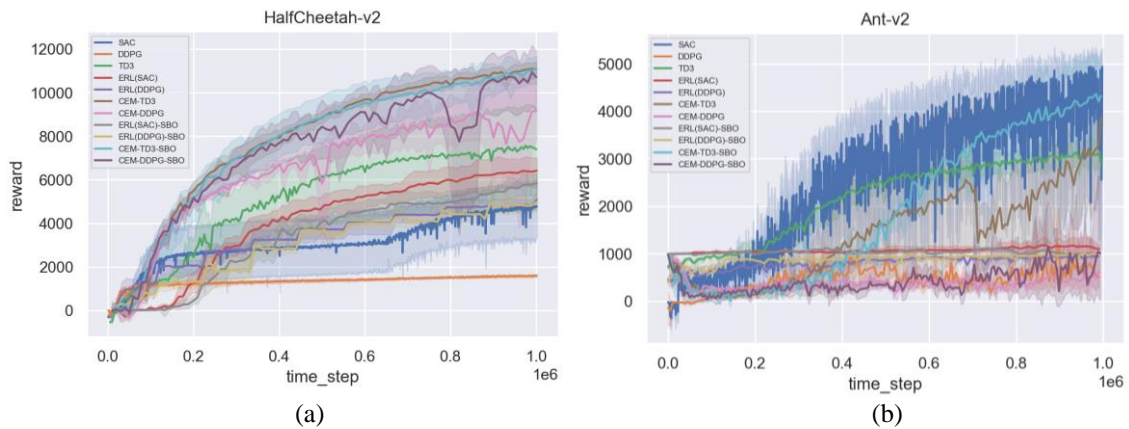


圖 10. HC (a)和 AT (b)

在 HC 環境中，ERL(SAC)-SBO 和 ERL(DDPG)-SBO 輸了原本方法，CEM-DDPG-SBO 贏了原本方法，而剩下加了多工的方法和原本的方法沒有顯著的差異。

在 AT 環境中，CEM-TD3-SBO 比原本 CEM-TD3 贏了很多，而剩下加了多工的方法和原本的方法沒有顯著的差異。

(2) HC 和 SW 組合

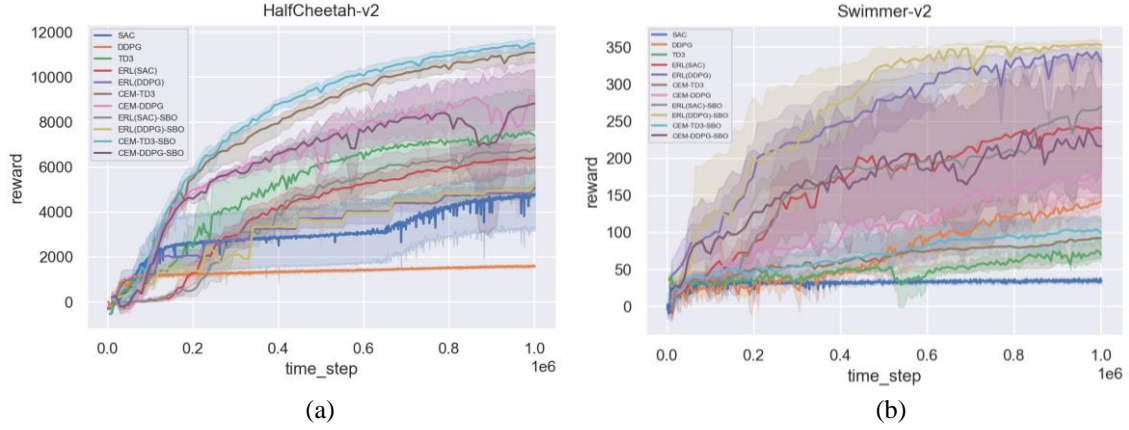


圖 11. HC (a)和 SW (b)

在 HC 環境中，而 CEM-TD3-SBO 比原本的 CEM-TD3 贏了一點，ERL(SAC)-SBO 到後面贏了 ERL(SAC)一點，而剩下加了多工的方法和原本的方法沒有顯著的差異。

在 SW 環境中，ERL(DDPG)-SBO 和 CEM-DDPG-SBO 贏了原本的方法，而剩下加了多工的方法和原本的方法沒有顯著的差異。

(3) HC 和 WK 組合

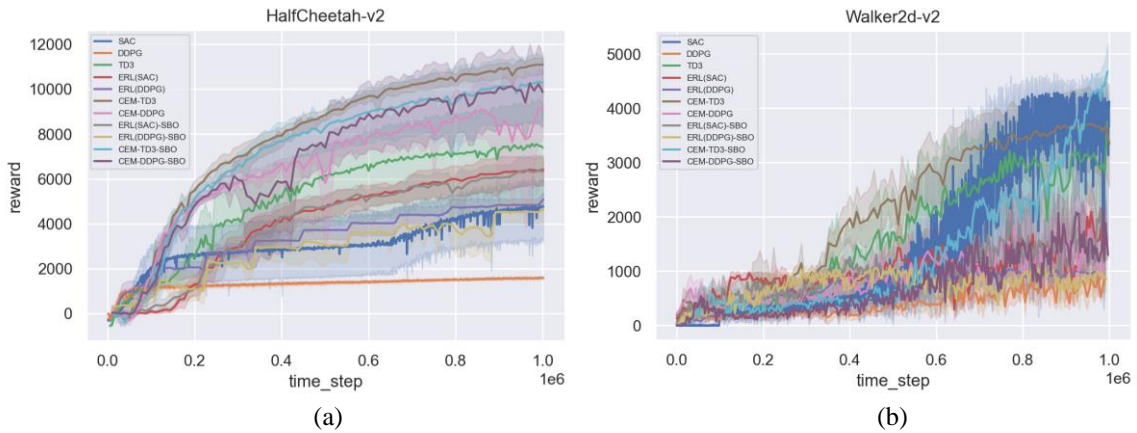


圖 12. HC (a)和 WK (b)

在 HC 環境中，CEM-DDPG-SBO 贏了原本方法，而 CEM-TD3-SBO 和 ERL(DDPG)輸了原本方法，剩下加了多工的方法都比原本的方法輸了一點的。

在 WK 環境中，CEM-TD3-SBO 是到後面才開始上升贏過原本的 CEM-TD3，而剩下加了多工的方法和原本的方法沒有顯著的差異。

(4) SW 和 AT 組合

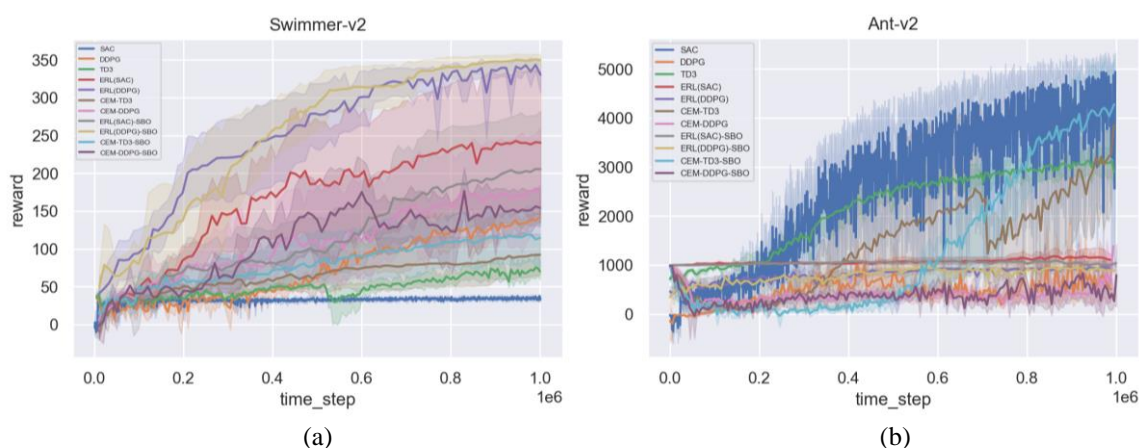


圖 13. SW (a)和 AT (b)

在 SW 環境中，CEM-TD3-SBO 比原本贏了一點，ERL(DDPG)-SBO 到後面有贏原本方法，CEM-DDPG-SBO 和原本 CEM-DDPG 則是在演化的過程中有輸有贏，ERL(SAC)-SBO 比原本差。

在 AT 環境中，CEM-TD3-SBO 到後面贏了 CEM-TD3，而剩下加了多工的方法和原本的方法沒有顯著的差異。

(5) SW 和 WK 組合

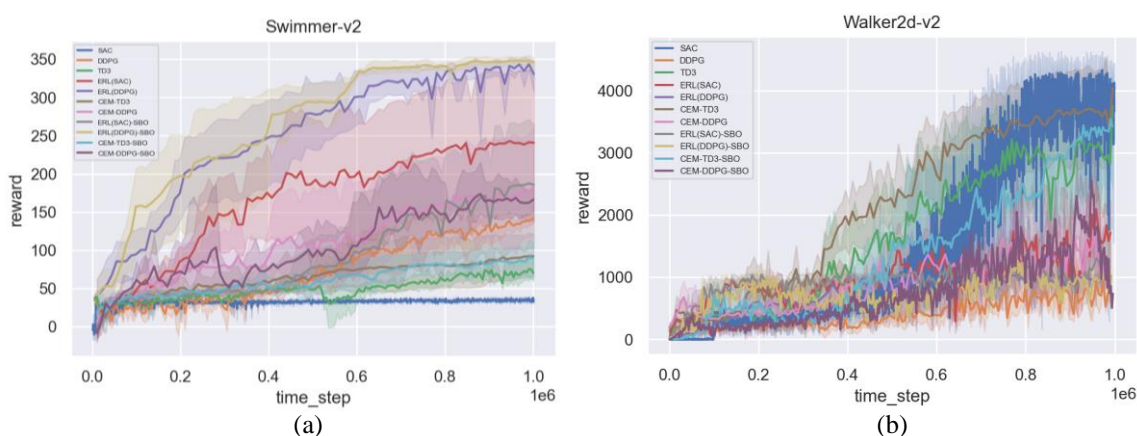


圖 14. SW (a)和 WK (b)

在 SW 環境中，ERL(DDPG)-SBO 贏了原本方法一點，ERL(SAC)-SBO 則比較差，而 CEM-DDPG-SBO 一開始較差但之後和原本的 CEM-DDPG 差不多，而剩下加了多工的方法和原本的方法沒有顯著的差異。

在 WK 環境中，CEM-TD3-SBO 和 ERL(SAC)-SBO 都比加了多工前差，而剩下加了多工的方法和原本的方法沒有顯著的差異。

(6) WK 和 AT 組合

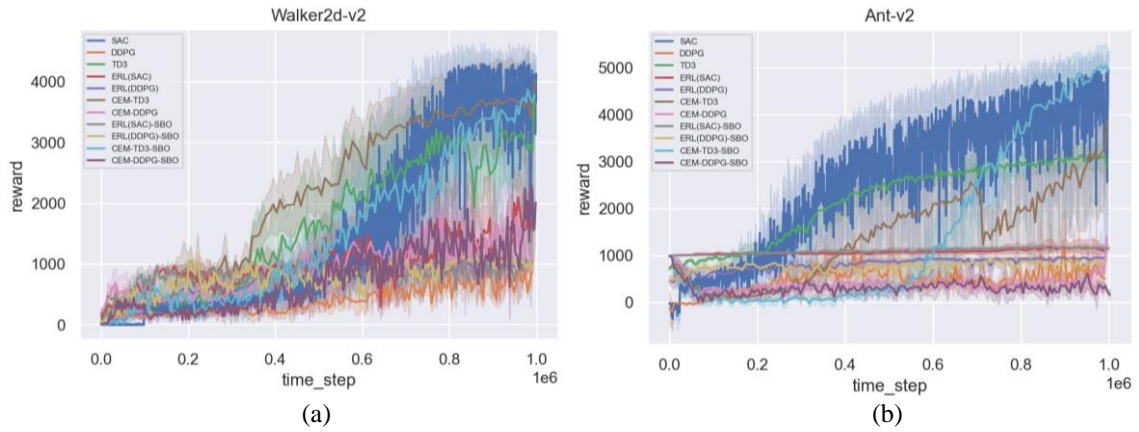


圖 15. WK (a)和 AT (b)

在 WK 環境中，CEM-TD3-SBO 和 ERL(DDPG)-SBO 比原本還差，而剩下加了多工的方法和原本的方法沒有顯著的差異。

在 AT 環境中，CEM-TD3-SBO 後面贏了原本的 CEM-TD3，而剩下加了多工的方法和原本的方法沒有顯著的差異。

六、結論與未來展望

6.1 討論

透過以上實驗數據的分析，我們發現確實是有組合能夠提升訓練速度，但也有的組合是維持進行演化多工前的表現甚至有些表現變得比原本還差，整體來說 CEM-TD3 的演算法在原本 ERL 和 CEM-RL 方法中在大多數的環境下都是表現最好的，而在最後一筆訓練數據的平均數來說 CEM-TD3-SBO 贏了 9 個加了多工前的 CEM-TD3 方法(表 4)，至於訓練效果維持原狀或是更差的情況，有可能是因為我們在不同維度間的任務交換個體時不考慮其相關關係採用的是直接對齊的方式等原因，這些都是值得我們去分析和討論的。

6.2 未來展望

1. 結合更多環境

目前因為電腦配置和 GPU 記憶體大小的問題，我們的演算法再加上 SBO 時，最多只能做兩個環境的多工訓練。但如果能一次多工處理更多環境的話，子代交換的情況就能更多元，實驗獲得的獎勵數據或許就能更高。

2. 優化演算法

我們目前專題使用的方法使用之 SBO 演算法在交換子代個體的步驟時，我們目前只有單純地把交換的個體模型維度對齊。但是每個環境個體模型所需要的參數維度不盡相同，我們並未考慮其中的相關性，也因為這樣，使得有些實驗結果的表現變得比原本未加入多工的方法還差，因此在未來也許在交換個體時可一進行 Mapping 的動作，能夠讓本專題所提出的方法結果更好。

3. 嘗試其他演化多工的演算法

因為我們這次的專題使用到了專題指導老師所提出之 SBO 演算法，但其實還有很多演化多工的演算法已經發表，也許我們可以嘗試結合其他演化多工演算法進行研究。

4. 嘗試其他不同的增強學習演算法結合演化多工的概念

這次的專題實驗，在深度增強學習演算法的部分，使用的是 DDPG、TD3、SAC，但是還有一些常被單獨用來訓練環境的深度增強學習演算法，尚未實驗過結合演化多工能達成的效果，例如：PPO、TRPO、A2C 等等，這也是未來可以嘗試的方向。

七、心得

本次專題發想題目之初，我們有意識到人工智慧的應用越來越多元，但並沒有明確的方向，在閱讀了一些論文以及與老師討論之下，我們選擇了深度增強學習作為我們專題的主軸。

大學期間裡，我們幾乎沒有學過深度增強學習的相關知識。以至於最初接觸畢業專題時，不知道該從何下手，可以說是完全的從零開始。老師了解情況後，推薦一些論文的網站給我們去查閱。漫長的使用裡，我們逐漸累積了找論文、判別哪些論文是與主題相關或契合主題的能力。在這期間，一邊閱讀論文，一邊尋找靈感，而最終我們選定用 CEM-RL 和 ERL 結合 SBO 後比較演算法的實驗。

有了方向後，緊接著就是閱讀畢業專題的相關論文，但由於基礎不足，我們也透過網路尋找相關的線上課程來學習。經歷了這一切，我們閱讀期刊論文變得更有效率，也深刻體認到自主學習的重要性。

之後，我們開始使用 OpenAI Gym。因為網路上的資源大多是國外的，我們也都只能讀國外的資料，再加上使用 OpenAI Gym 會用到其他套件模組，而這些套件模組又多個不同的版本，導致版本不相容使原始碼無法運行，也導致在這期間花了很多時間在試錯，所幸最後我們還是有把程式碼跑起來，並成功使用 OpenAI Gym 的功能。

接下來，就是選擇演算法和改程式碼。在這段期間，我們學到了如何去過濾網路上眾多的資料，經過不斷嘗試與觀察演算法的效果，我們才決定使用 CEM-RL 和 ERL 作為我們專題的主要研究。後來，在老師的建議之下，我們使用 SBO 將 CEM-RL 和 ERL 做多工處理。過程中，我們除了程式碼的套件不相符而多次卡關，也在理解論文所提供之原始碼下了不少功夫，但最終也在一次次嘗試和學習中解決。

在透過上述種種製作畢業專題的心得，我們在這過程中，真的學到很多東西。學到了如何去挑選合適的論文題材，以及如何去解決問題。在製作畢業專題的過程，很多東西都是我們在大學期間不會學到的。在課堂間也許有老師的幫助，同學之間做作業的題目也都相同，因此我們可以很快地解決問題。但是畢業專題是必須整合大學所學之外，再加上其他知識才能做出來，會比一般專題所花的精力和時間還多，同時，這也算是我們第一次做的一個大型的專題，肯定不是一下子就能做到的，所以我們也學到了如何去排定日期、分工合作，才能按部就班地將專題製作完成。

也很感謝老師讓我們在製作畢業專題期間，從一開始就一步步讓我們了解未來不管是研究所或是業界會對我們的要求。尤其是每周的開會討論，我們學士班的會跟碩班的學長一起開會，也因為這樣，我們每周都可以聽到不同論文的報告，也更了解碩班對於同個主題，又會有怎樣的見解。另外，也很感謝老師不只在畢業專題之外對我們的提點，像是在製作簡報怎麼做會比較好、報期刊論文時，如何講會比較好等諸如此類，同時也謝謝指導老師的實驗室提供機器給我們做專題，因為我們的專題所需要的計算資源較大。

八、參考資料

- [1] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.
- [2] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290, 2018.
- [3] Scott Fujimoto, Herke van Hoof, and Dave Meger. Addressing function approximation error in actor-critic methods. CoRR, abs/1802.09477, 2018.
- [4] Eiben, A. E., and Smith, J. E. 2003. Introduction to Evolutionary Computing. Natural Computing. Springer-Verlag.
- [5] Khadka S. and Tumer K., “Evolution-guided policy gradient in reinforcement learning,” in *NeurIPS*, 2018.
- [6] Pourchot A. and Sigaud O., “CEM-RL: Combining evolutionary and gradient-based methods for policy search,” in *ICLR*, 2019.
- [7] R.-T. Liaw and C.-K. Ting, “Evolutionary manytasking optimization based on symbiosis in biocoenosis,” in Thirty-Third AAAI Conference on Artificial Intelligence, 2019.

附錄

A. 安裝 Anaconda

1. 進入 Anaconda 官網 <https://www.anaconda.com/products/distribution>，因本專題的作業系統是 Ubuntu，所以以 Linux 為例。下載下來後，開啟 Terminal，路徑設定到下載檔案的資料夾，輸入命令後 `bash Anaconda3-2022.10-Linux-x86_64.sh` 即完成安裝



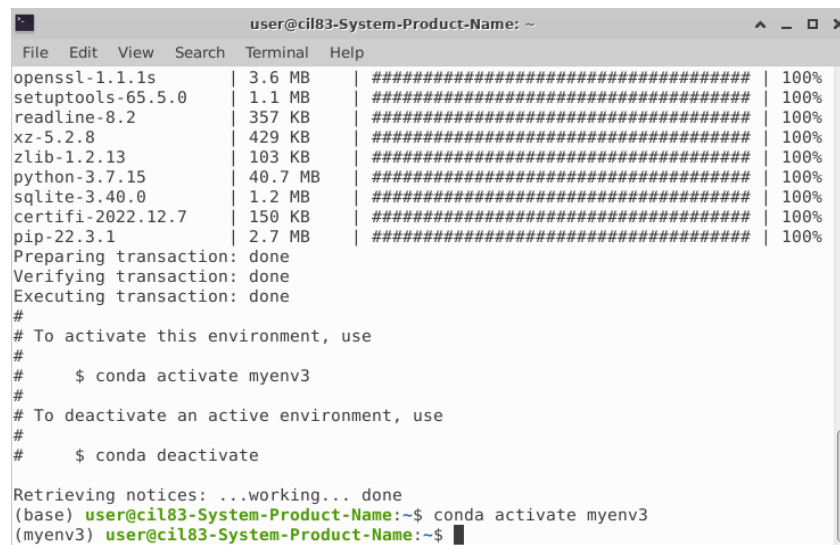
圖 16. 從官網下載 Anaconda 安裝包

2. 用 conda 建立及管理 python 虛擬環境，下指令 `conda create --name [環境名稱] python=3.7`，這裡 python 版本以 3.7 為例。

```
user@cil83-System-Product-Name: ~  
File Edit View Search Terminal Help  
(base) user@cil83-System-Product-Name:~$ conda create --name myenv3 python=3.7  
Collecting package metadata (current_repodata.json): done  
Solving environment: done  
  
==> WARNING: A newer version of conda exists. <==  
current version: 22.9.0  
latest version: 22.11.1  
  
Please update conda by running  
  
$ conda update -n base -c defaults conda  
  
## Package Plan ##  
  
environment location: /home/user/anaconda3/envs/myenv3  
  
added / updated specs:  
- python=3.7  
  
The following packages will be downloaded:
```

圖 17. 建立 python 虛擬環境

3. 利用指令 `conda activate` [環境名稱] 喚醒剛剛建立好的虛擬環境

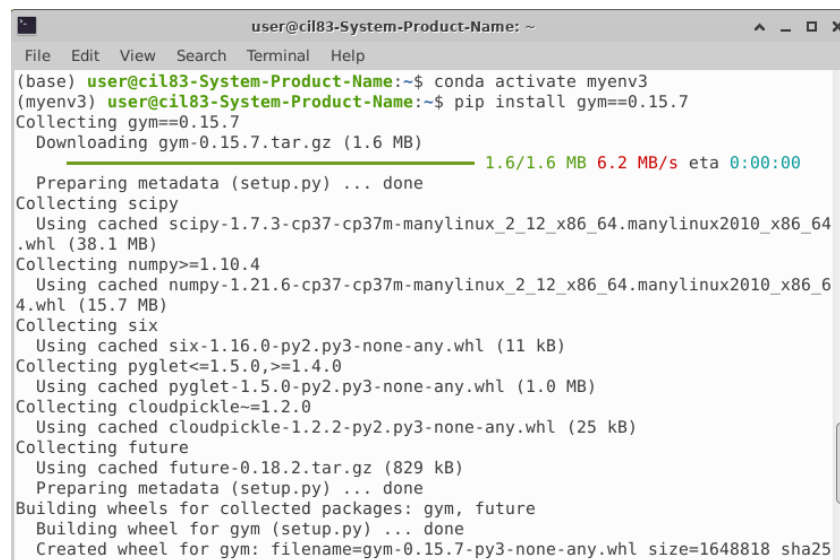


```
user@cil83-System-Product-Name: ~  
File Edit View Search Terminal Help  
openssl-1.1.1s | 3.6 MB | ##### | 100%  
setuptools-65.5.0 | 1.1 MB | ##### | 100%  
readline-8.2 | 357 KB | ##### | 100%  
xz-5.2.8 | 429 KB | ##### | 100%  
zlib-1.2.13 | 103 KB | ##### | 100%  
python-3.7.15 | 40.7 MB | ##### | 100%  
sqlite-3.40.0 | 1.2 MB | ##### | 100%  
certifi-2022.12.7 | 150 KB | ##### | 100%  
pip-22.3.1 | 2.7 MB | ##### | 100%  
Preparing transaction: done  
Verifying transaction: done  
Executing transaction: done  
#  
# To activate this environment, use  
#  
# $ conda activate myenv3  
#  
# To deactivate an active environment, use  
#  
# $ conda deactivate  
  
Retrieving notices: ...working... done  
(base) user@cil83-System-Product-Name:~$ conda activate myenv3  
(myenv3) user@cil83-System-Product-Name:~$
```

圖 18. 激活虛擬環境

B. 安裝 Openai gym

在 Terminal 中輸入指令 `pip install gym`，後面可以加上需要的版本



```
user@cil83-System-Product-Name: ~  
File Edit View Search Terminal Help  
(base) user@cil83-System-Product-Name:~$ conda activate myenv3  
(myenv3) user@cil83-System-Product-Name:~$ pip install gym==0.15.7  
Collecting gym==0.15.7  
  Downloading gym-0.15.7.tar.gz (1.6 MB)  
    1.6/1.6 MB 6.2 MB/s eta 0:00:00  
  Preparing metadata (setup.py) ... done  
Collecting scipy  
  Using cached scipy-1.7.3-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (38.1 MB)  
Collecting numpy>=1.10.4  
  Using cached numpy-1.21.6-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (15.7 MB)  
Collecting six  
  Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)  
Collecting pygamelet<=1.5.0,>=1.4.0  
  Using cached pygamelet-1.5.0-py2.py3-none-any.whl (1.0 MB)  
Collecting cloudpickle~=1.2.0  
  Using cached cloudpickle-1.2.2-py2.py3-none-any.whl (25 kB)  
Collecting future  
  Using cached future-0.18.2.tar.gz (829 kB)  
  Preparing metadata (setup.py) ... done  
Building wheels for collected packages: gym, future  
Building wheel for gym (setup.py) ... done  
Created wheel for gym: filename=gym-0.15.7-py3-none-any.whl size=1648818 sha256
```

圖 19. 安裝 openai gym

C. 安裝 MuJoCo

1. 到 MuJoCo 的 Github 官網 <https://github.com/openai/mujoco-py> 下載官方的 MuJoCo 安裝包，將下載下來的安裝包解壓縮放入 `~/mujoco/mujoco210` 資料夾中

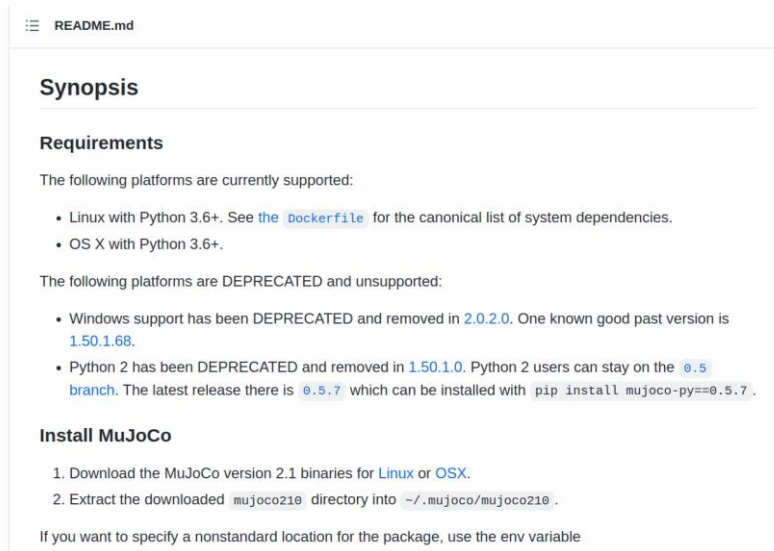


圖 20. 下載 MuJoCo 安裝包

2. 在 Terminal 中輸入指令 `pip3 install -U 'mujoco-py<2.2,>=2.1'`

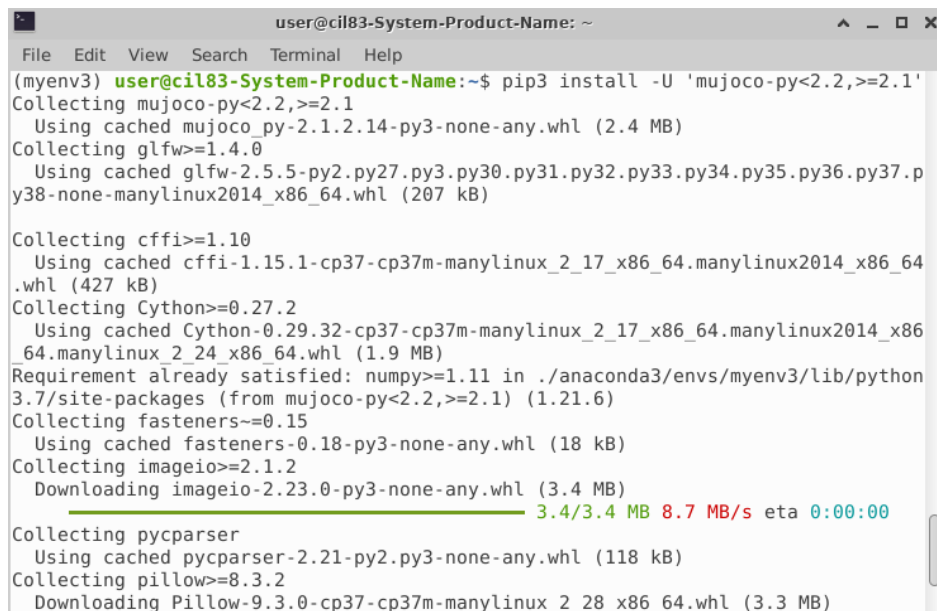


圖 21. 安裝 MuJoCo

3. 在 Terminal 中輸入指令 `gedit ~/.bashrc` 添加環境變數，`user_name` 請輸入自己電腦的使用者名稱
- ```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/user_name/.mujoco/mujoco210/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/nvidia
export LD_PRELOAD=/usr/lib/x86_64-linux-gnu/libGLEW.so
```
- 添加完畢後在 Terminal 中輸入指令 `source ~/.bashrc`
4. 安裝完成後用官方給的程式碼測試是否安裝成功

```
import mujoco_py
import os
mj_path = mujoco_py.utils.discover_mujoco()
xml_path = os.path.join(mj_path, 'model', 'humanoid.xml')
model = mujoco_py.load_model_from_path(xml_path)
sim = mujoco_py.MjSim(model)

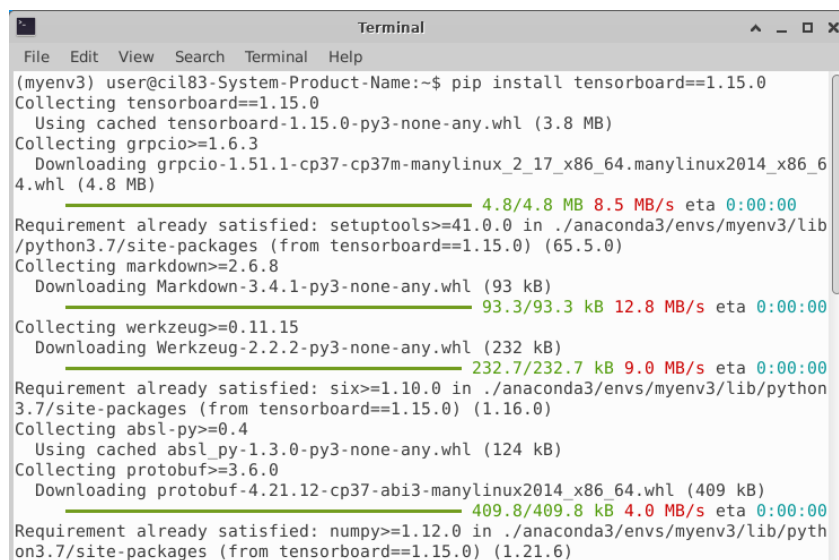
print(sim.data.qpos)
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

sim.step()
print(sim.data.qpos)
[-2.09531783e-19 2.72130735e-05 6.14480786e-22 -3.45474715e-06
7.42993721e-06 -1.40711141e-04 -3.04253586e-04 -2.07559344e-04
8.50646247e-05 -3.45474715e-06 7.42993721e-06 -1.40711141e-04
-3.04253586e-04 -2.07559344e-04 -8.50646247e-05 1.11317030e-04
-7.03465386e-05 -2.22862221e-05 -1.11317030e-04 7.03465386e-05
-2.22862221e-05]
```

圖 22. 測試是否安裝成功之程式碼

## D. 安裝其他第三方套件模組

本專題有用到 Pytorch、Numpy、Tensorboard、Fastrand 等其他第三方套件模組，這些東西都利用 `pip install` 加上版本和模組名稱去做安裝。



```
Terminal
File Edit View Search Terminal Help
(myenv3) user@cil83-System-Product-Name:~$ pip install tensorboard==1.15.0
Collecting tensorboard==1.15.0
 Using cached tensorboard-1.15.0-py3-none-any.whl (3.8 MB)
Collecting grpcio==1.6.3
 Downloading grpcio-1.51.1-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.8 MB)
 4.8/4.8 MB 8.5 MB/s eta 0:00:00
Requirement already satisfied: setuptools==41.0.0 in ./anaconda3/envs/myenv3/lib/python3.7/site-packages (from tensorboard==1.15.0) (65.5.0)
Collecting markdown==2.6.8
 Downloading Markdown-3.4.1-py3-none-any.whl (93 kB)
 93.3/93.3 kB 12.8 MB/s eta 0:00:00
Collecting werkzeug==0.11.15
 Downloading Werkzeug-2.2.2-py3-none-any.whl (232 kB)
 232.7/232.7 kB 9.0 MB/s eta 0:00:00
Requirement already satisfied: six>=1.10.0 in ./anaconda3/envs/myenv3/lib/python3.7/site-packages (from tensorboard==1.15.0) (1.16.0)
Collecting absl-py==0.4
 Using cached absl_py-1.3.0-py3-none-any.whl (124 kB)
Collecting protobuf==3.6.0
 Downloading protobuf-4.21.12-cp37-abi3-manylinux2014_x86_64.whl (409 kB)
 409.8/409.8 kB 4.0 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.12.0 in ./anaconda3/envs/myenv3/lib/python3.7/site-packages (from tensorboard==1.15.0) (1.21.6)
```

圖 23. 安裝 Tensorboard 套件模組