以逸待勞-強化學習訓練一動不如一靜

Constant in an Ever-Changing World

吳建中, 林俊成, 黃月華, 廖容佐
輔仁大學資訊工程學系
E-mail: andywu.academic@gmail.com, cclin@csie.fju.edu.tw,
yhhuang@csie.fju.edu.tw, rtliaw@csie.fju.edu.tw

https://github.com/AndyWu101/CIC

摘要

強化學習的訓練過程常伴隨劇烈的震蕩。導致演算法的不穩定性與性能下降。本文提出了一種以逸待勞的架構 (CIC)，增強演算法的穩定性以提升效能。CIC 有代表策略以及當前策略。CIC 不盲目變動代表策略，而是有選擇的在當前策略更優時更新代表策略。CIC 使用一種自適應調整的機制，使代表策略與當前策略共同幫助 critic 訓練。我們分別在 MuJoCo 的 5 個環境上測試了 CIC 的表現。結果顯示 CIC 可以在不增加計算成本的情況下提升傳統演算法的性能。

關鍵字: 強化學習、Actor-Critic、連續控制任務

1. 緒論

強化學習 (Reinforcement Learning) 被研究於搜尋遊戲最佳策略、機械控制、訓練語言模型等方面並展現強化學習的應用價值。依照算法性質，強化學習可分為以下兩種類別: 1) 基於價值的方法 (Value-based Approach) 2) 基於策略的方法 (Policy-based Approach)。基於策略的方法中 Actor-Critic 是一種在連續控制任務中非常有效的架構。本次研究主要聚焦於 Actor-Critic 方法。

在研究中我們發現，Actor-Critic 架構中的 actor 性能在訓練中有時候會突然大幅下降。然而傳統 Actor-Critic 架構中 actor 和 critic 互為倚仗，如果某方性能下降，可能導致惡性循環，加劇演算法的不穩定性。

為了解決這個問題，本文提出了一種「以逸待勞」的新架構 (CIC)，其在不增加計算成本的情況下，通過設計打斷惡性循環，增強演算法穩定性。實驗結果顯示 CIC 在 4 種演算法上都有一定程度的性能提升。

2. 準備工作

為了系統化地描述強化學習的過程，通常將強化學習問題建模為馬可夫決策過程 (MDP[1])。MDP 提供了一個數學框架，透過 5 元組 $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma >$ 定義，其中:

- $\mathcal{S}$ 是狀態空間
- $\mathcal{A}$ 是動作空間
- $\mathcal{R}(s,a) = \mathbb{E}\big[r_t \mid s_t = s, a_t = a\big]$ 是回饋函數
- $\mathcal{P}(s_{t+1} \mid s_t, a_t)$ 是指從狀態 s_t 採取動作 a_t 後環境狀態轉移成 s_{t+1} 的機率分布
- $\gamma \in [0,1]$ 是折扣因子，代表對於未來回報的重視程度

在強化學習中，actor 在每個離散時間步 t 接收環境給予的狀態 $s_t \in \mathcal{S}$ 並依據其策略 π 選擇動作 $a_t \sim \pi(\cdot \mid s_t)$，獲得環境的回饋 $\mathcal{R}(s_t, a_t)$。actor 的目標是學習最佳策略獲得最大的總折扣回報

$$R_t = \sum_{i=t}^{T} \gamma^{i-t} \mathcal{R}(s_t, a_t)$$

對於給定 π，可以定義狀態-動作價值函數 (或稱 Q 函數)

$$Q^\pi(s_t, a_t) = \mathbb{E}\Big[R_t \;\Big|\; s_t, a_t\Big]$$

同時 Q 函數滿足 Bellman Expectation Equation[1]:

$$Q^\pi(s_t, a_t) = \mathbb{E}\Big[\mathcal{R}(s_t, a_t) + \gamma \mathbb{E}[Q^\pi(s_{t+1}, a_{t+1})]\Big]$$

Actor-Critic 的方法中:

Critic 的損失函數定義為最小化

$$J_{Q^\pi} = \mathbb{E}\Big[Q^\pi(s_t, a_t) - y_t\Big],$$

$$y_t = \mathcal{R}(s_t, a_t) + \gamma Q^\pi(s_{t+1}, a_{t+1})$$

Actor 的目標函數定義為最小化

$$J_\pi = \mathbb{E}\Big[- Q^\pi(s_t, a_t) \;\Big|\; a_t \sim \pi(\cdot \mid s_t)\Big]$$

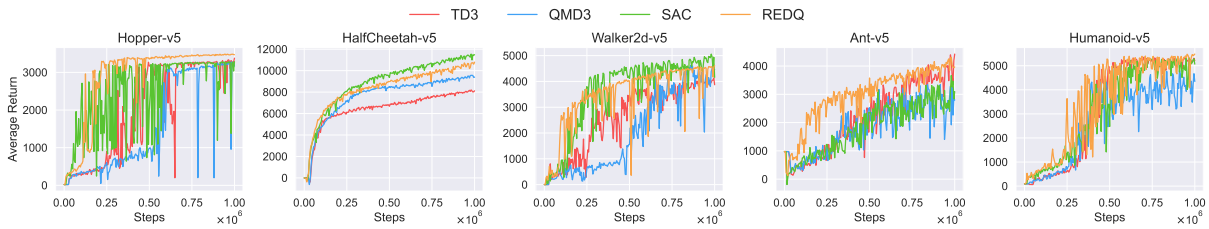

Figure 1: Actor-Critic 演算法的不穩定性

3. 相關研究

在連續控制任務中著名算法 DDPG[2] 採用 Actor-Critic 方法，使用單一 critic 估計 Q 值以及單一確定性策略的 actor，並且提出軟更新 (sort update) 的概念，後續成為強化學習算法中常見的技巧。因此 DDPG 在連續控制任務中具有承先啟後的意義。

然而 DDPG 由於訓練時 Q 值高估導致的不穩定導致收斂不佳。為解決此問題，TD3[3] 引入兩個 critic 取最小值以估計目標 Q 值，減緩 Q 值高估問題，同時為 actor 引入 actor target 以及延遲更新，增加訓練過程的穩定性。

TD3 由於估計 Q 值時使用最小值操作，導致 Q 值低估的現象抑制了 actor 的探索，為了改善此現象，QMD3[4] 提出使用 N 個 critic (推薦 $N = 4$)，將每個 critic 估計的 Q 值排序後取第 $\lfloor \frac{N}{2} \rfloor$ 個 Q 值作為最終估計，能夠緩解 Q 值高估與低估的發生。

與 TD3 同一時期 SAC[5] 一樣使用兩個 critic 取最小值以估計目標 Q 值，不同的是 SAC 採用隨機策略的 actor 以及基於 soft Q 函數 [6] 訓練 critic，soft Q 函數除了原有的環境回饋之外引入動作的熵，如此 critic 擁有引導 actor 嘗試更多樣動作的能力。

後續的研究中，REDQ[7] 在 SAC 的基礎上使用 N 個 critic (推薦 $N = 10$)。具體而言，在訓練 critic 時，REDQ 每次從 N 個 critic 中隨機抽選 2 個計算目標 Q 值；在訓練 actor 時，REDQ 使用 N 個 critic 的平均梯度訓練 actor。以此增強穩定性，進一步提升 SAC 的效能。

4. 穩定性分析

Figure 1 中我們展示了 4 種 Actor-Critic 演算法在 5 個環境中單次訓練的結果，可以看到除了 HalfCheetah-v5 以外，其他環境的訓練過程皆相當震蕩，有時甚至會從接近滿分瞬間掉到接近 0 分。

5. 方法

本研究提出了一個新穎的 Actor-Critic 架構，Figure 2展示了 CIC 和傳統演算法的架構差異。CIC 通過以逸待勞以及自適應調整 λ 機制，改善了傳統演算法的穩定性。Algorithm 1描述了演算法流程，詳細介紹如下。

5.1 以逸待勞

在傳統方法中，actor 或 actor target 都會無條件的進行改變，哪怕改變會導致 actor 或 actor target 的性能下降。為了解決這個問題，CIC 的架構有兩個 actor。actor1 為高分 actor，其不會受到訓練或任何改變，用以維持穩定性；actor2 為接受訓

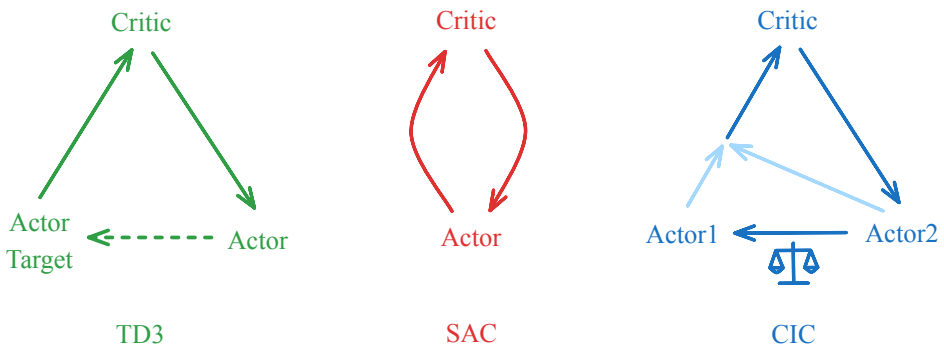

Figure 2: CIC 架構圖

Algorithm 1 CIC

Initialize actor1 π_{ϕ_1} with random parameters ϕ_1
Initialize actor2 $\pi_{\phi_2} \leftarrow \pi_{\phi_1}$
Initialize critic networks Q_{θ_i} with random parameters θ_i for $i \in \{1 \cdots q\}$
Initialize critic target networks $\theta'_i \leftarrow \theta_i$ for $i \in \{1 \cdots q\}$
$t \leftarrow$ Initialize replay buffer $\mathcal{R}$
$\lambda \leftarrow 0$
Fill lambda buffer Λ up with $(\lambda, -\infty)$

while $t < T$ **do**
    $score_1, steps_1 \leftarrow$ Evaluate($\pi_{\phi_1}, \mathcal{R}$) ▷ Store transitions (s, a, r, s') in $\mathcal{R}$
    $score_2, steps_2 \leftarrow$ Evaluate($\pi_{\phi_2}, \mathcal{R}$) ▷ Store transitions (s, a, r, s') in $\mathcal{R}$
    $\pi^{\mathcal{S}}_{\phi_1} \leftarrow \pi^{\mathcal{S}}_{\phi_1} \cup score_1$
    $\pi^{\mathcal{S}}_{\phi_2} \leftarrow score_2$
    $\Delta t \leftarrow steps_1 + steps_2$
    Replace the oldest pair of Λ with $(\lambda, score_2)$

    if $\pi^{\bar{\mathcal{S}}}_{\phi_1} < \pi^{\bar{\mathcal{S}}}_{\phi_2}$ **then**
        $\pi_{\phi_1} \leftarrow \pi_{\phi_2}$
    end if

    $\lambda \leftarrow \text{mean}\{\lambda_i \mid (\lambda_i, score_i) \in \text{TopHalf}_{score}(\Lambda)\}$
    $\lambda \leftarrow \text{clip}(\lambda + \epsilon, 0, 1),\ \epsilon \sim \mathcal{N}(0, \sigma)$
    for $i \leftarrow 1$ to Δt **do**
        $\mathcal{B} \leftarrow$ Sample a mini-batch of N transitions (s, a, r, s') from $\mathcal{R}$
        for $j \leftarrow 1$ to N **do**
            $(s, a, r, s') \leftarrow \mathcal{B}_j$
            if $j \leq \lfloor N \cdot \lambda \rfloor$ **then**
                $a' \leftarrow \pi_{\phi_1}(s')$
            else
                $a' \leftarrow \pi_{\phi_2}(s')$
            end if
            $\mathcal{B}_j \leftarrow (s, a, r, s', a')$
        end for
        Train critics $Q_{\theta_{i \cdots q}}$ and actor2 π_{ϕ_2} by $\mathcal{B}$
        Update critic targets $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
    end for
    $t \leftarrow t + \Delta t$
end while
return π_{ϕ_1}

練的 actor，用以探索並超越 actor1，使演算法進步。

為了獲知 actor 的性能，我們將訓練流程從互動一步訓練一步，改成互動一局訓練一局。由於 actor1 不會改變，所以我們持續紀錄其歷史得分；然而 actor2 每輪都會受到梯度訓練並改變參數，所以只使用當前得分。當 actor2 的分數高於 actor1，則 actor2 會成為新的 actor1。

在設計中我們認為 actor1 的分數將於一定局數後逐漸收斂，同時為了避免失去動作多樣性，所以我們限制 actor1 與環境互動的最高局數為 10 局。另外由於環境和策略本身的隨機性，有時需要玩更多局以獲知 actor 的真正性能。如果只玩一局，容易因為運氣獲得高分導致 actor2 的得分高

於 actor1。因此 CIC 使用參數 κ 允許 actor2 起始評估超過一局，以多局分數作為基準，確保 actor2 分數的可信度。

5.2 λ 與其自適應調整機制

CIC 不使用 actor target，而是使用 actor1 和 actor2 共同幫助 critic 訓練，並通過一個自適應調整的係數 λ 控制 actor1 的參與比例，這使得 critic 的訓練更穩定，從而打斷惡性循環。具體來說，CIC 訓練 critic 時透過 λ 控制一定比例的 mini-batch 由 actor1 決定 a'，剩餘由 actor2 決定。如此 critic 將同時引入 actor1 與 actor2 的知識。

λ 在不同環境以及不同訓練階段有不同的最佳值。因此 CIC 設計了一個機制自適應調整 λ。CIC 使用一個緩衝區 Λ 紀錄過去一段時間的 λ 與同時期 actor2 的分數。每次開始訓練前，從 Λ 中取分數最高的一半，計算這些 λ 的平均值，再加上一個常態擾動探索新的 λ，同時確保 $0 \leq \lambda \leq 1$。獲得 actor2 分數後，將 Λ 中最舊的紀錄替換為當前 $(\lambda, \text{actor2 分數})$。如此 CIC 能夠透過訓練過程的資訊自適應調整 λ。

6. 實驗

6.1 MuJoCo

MuJoCo[8] 全名為 Multi-Joint dynamics with Contact，是一款物理引擎，主要用於提供真實的物理模擬，適用於需要快速且精準模擬的場合。其特色在於能夠同時兼顧物理精確性與計算效率，特別針對機器人與環境之間的物理接觸進行建模與模擬。本研究採用官方最新的版本 v5。https://github.com/Farama-Foundation/Gymnasium

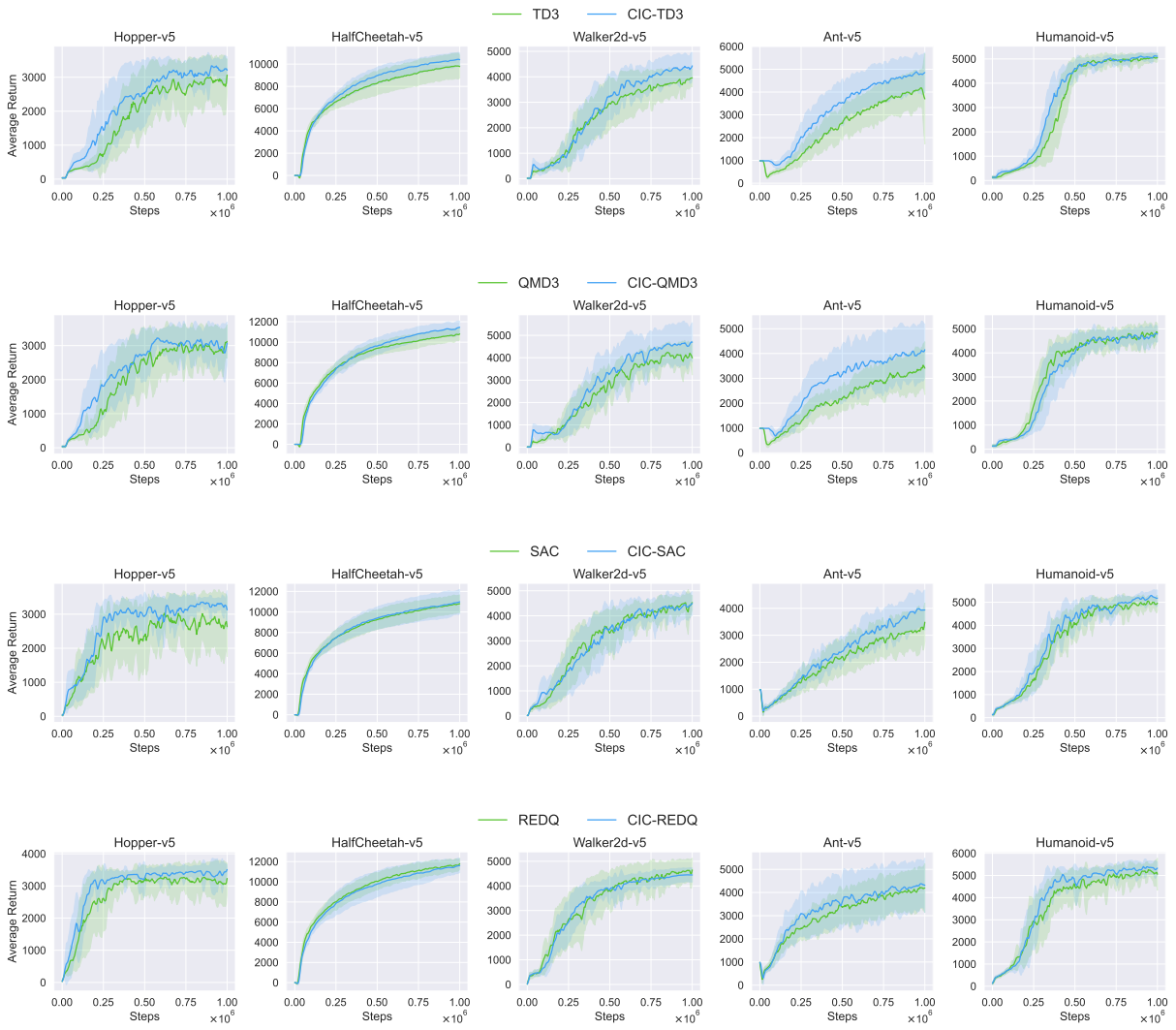

Figure 3: CIC 對比實驗

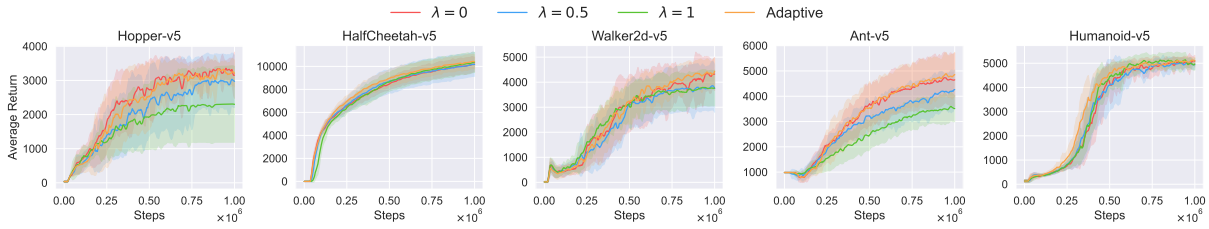

Figure 4: CIC-TD3 固定 λ 分析實驗

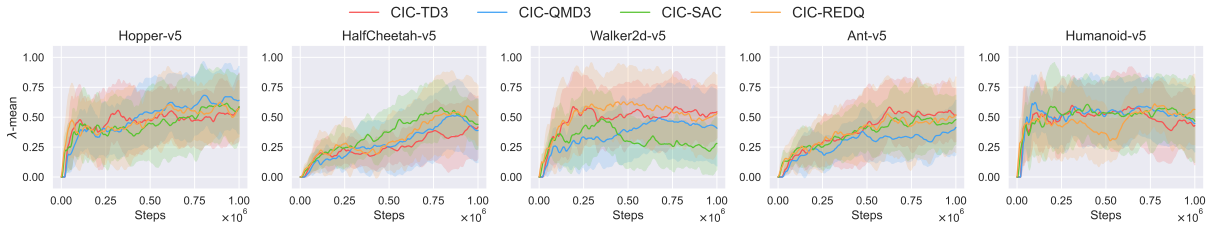

Figure 5: CIC λ 隨訓練過程自適應調整變化

6.2 實驗設定

我們在 MuJoCo 的 5 個環境上 (Hopper, HalfCheetah, Walker2d, Ant, Humanoid) 測試演算法性能。我們每 5000 步以 actor 20 局的平均性能作為基準。學習曲線由 10 個 seed 的平均組成，陰影部分為正負一倍標準差，曲線經過平滑化。Table 2 展示了實驗的超參數設定。

6.3 實驗結果

Figure 3 顯示，CIC-TD3 在 Hopper-v5、HalfCheetah-v5、Walker2d-v5、Ant-v5 上有明顯的效能提升，在 Humanoid-v5 有較快的收斂速度。CIC-QMD3 在 Hopper-v5 上收斂較快，在 HalfCheetah-v5、Walker2d-v5、Ant-v5 上有明顯的效能提升，但在 Humanoid-v5 上收斂較慢。CIC-SAC 在 Hopper-v5、Ant-v5 上有明顯的效能提升，其他環境則維持相同表現。CIC-REDQ 在 Hopper-v5、Ant-v5、Humanoid-v5 上效能有一定提升，在 Walker2d-v5 上維持相同表現，但在 HalfCheetah-v5 上收斂略慢。Figure 1中可見 Hopper-v5 是一個訓練過程特別震蕩的環境，但在 Hopper-v5 上 CIC 的標準差都比原演算法低，顯示了其增強穩定性的效果。

Figure 4 顯示，在 5 個環境中自適應 λ 都可以獲得最佳性能，固定 $\lambda = 0$ 也有較佳表現，但固定 $\lambda = 1$ 則會有較明顯的性能下降，代表策略相差太大時，不能完全依賴 actor1。Figure 5 顯示，大部分 λ 都會收斂在 0.5，但相較於 Figure 4 中的固定 $\lambda = 0.5$，自適應 λ 會有較佳表現，代表自適應機制是有效且必要的。Table 1 顯示，CIC 在 4 種演算法應用於 5 種環境，共 20 種情況下，其平均回撤都有大幅下降，進一步證明了 CIC 對穩定性的提升。

7. 結論

我們發現強化學習訓練過程常伴隨較大的不穩定震蕩，這降低了演算法的可靠性。因此本研究提出了 CIC，利用 2 個功能不同的 actor，配合自適應調整機制，提升了穩定性。結果表明強化學習訓練「一動不如一靜」，以逸待勞或許是更高明的做法。而且由於 CIC 的機制簡潔，因此可以輕鬆地將其添加到任何 Actor-Critic 演算法中。

Table 1: 各演算法平均回撤

	TD3	CIC-TD3	QMD3	CIC-QMD3	SAC	CIC-SAC	REDQ	CIC-REDQ
Hopper-v5	130 ± 42	$\mathbf{66 \pm 30}$	145 ± 64	$\mathbf{48 \pm 26}$	235 ± 95	$\mathbf{88 \pm 30}$	100 ± 65	$\mathbf{31 \pm 30}$
HalfCheetah-v5	71 ± 28	$\mathbf{32 \pm 8}$	71 ± 19	$\mathbf{20 \pm 7}$	67 ± 22	$\mathbf{25 \pm 7}$	86 ± 20	$\mathbf{19 \pm 5}$
Walker2d-v5	107 ± 24	$\mathbf{91 \pm 25}$	181 ± 31	$\mathbf{85 \pm 24}$	170 ± 22	$\mathbf{125 \pm 45}$	122 ± 35	$\mathbf{34 \pm 21}$
Ant-v5	137 ± 23	$\mathbf{88 \pm 23}$	176 ± 21	$\mathbf{86 \pm 16}$	155 ± 25	$\mathbf{140 \pm 11}$	185 ± 31	$\mathbf{85 \pm 17}$
Humanoid-v5	96 ± 20	$\mathbf{65 \pm 11}$	181 ± 27	$\mathbf{75 \pm 18}$	198 ± 24	$\mathbf{113 \pm 21}$	203 ± 56	$\mathbf{74 \pm 27}$

Table 2: 超參數設定

Hyper-parameter	TD3	QMD3	SAC	REDQ
Number of Critics (q)	2	4	2	10
Discount Factor (γ)	0.99	0.99	0.99	0.99
Learning Rate	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$
Optimizer	Adam	Adam	Adam	Adam
Batch Size	256	256	256	256
Actor Target	○	○	×	×
Critic Target	○	○	○	○
Soft Update Ratio (τ)	$5 \cdot 10^{-3}$	$5 \cdot 10^{-3}$	$5 \cdot 10^{-3}$	$5 \cdot 10^{-3}$
UTD Ratio	1	1	1	1
Delay Frequency	2	2	×	×
Warmup Steps	25000	25000	10000	5000
Exploration Noise	$\mathcal{N}(0, 0.1)$	$\mathcal{N}(0, 0.1)$	×	×
Target Policy Noise	$\mathcal{N}(0, 0.2)$	$\mathcal{N}(0, 0.2)$	×	×
Policy Noise Clip	$[-0.5, 0.5]$	$[-0.5, 0.5]$	×	×
Temperature (α)	×	×	Adaptive	Adaptive
Target Entropy	×	×	$-\lvert\mathcal{A}\rvert$	$\{-4, -3, -2, -1\}$
Log Std Clip	×	×	$[-20, 2]$	$[-20, 2]$
Ensemble Subset Size	×	×	×	2
	CIC-TD3	**CIC-QMD3**	**CIC-SAC**	**CIC-REDQ**
Actor2 Evaluations (κ)	1	2	1	2
Lambda Buffer Size ($\lvert\Lambda\rvert$)	10	10	10	6
Lambda Std. (σ)	0.1	0.1	0.1	0.1

8. 致謝

本研究由國家科學及技術委員會贊助－計畫編號 NSTC112-2221-E-030-008-MY3。

參考文獻

[1] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

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

[3] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[4] W. Wei, Y. Zhang, J. Liang, L. Li, and Y. Li, "Controlling underestimation bias in reinforcement learning via quasi-median operation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 36, no. 8, 2022, pp. 8621–8628.

[5] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. Pmlr, 2018, pp. 1861–1870.

[6] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *International conference on machine learning*. PMLR, 2017, pp. 1352–1361.

[7] X. Chen, C. Wang, Z. Zhou, and K. Ross, "Randomized ensembled double q-learning: Learning fast without a model," *arXiv preprint arXiv:2101.05982*, 2021.

[8] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.