深度學習作業報告

3D 模擬環境中的導航訓練

系級：
學號：
姓名：

2020 年 6 月 16 日

介紹

近期利用模擬環境來訓練神經網路非常盛行，本作業利用 Habitat-API(註 3)來獲得 3D 環境，並利用強化學習(Reinforcement Learning)方法，讓 Agent(代理人)進行環境導航。

本作業實作了兩種 RL 的演算法 PPO 和 Split+PPO，Split+PPO 為參考論文(註 7) 來實現，並比較了兩者之間的結果。

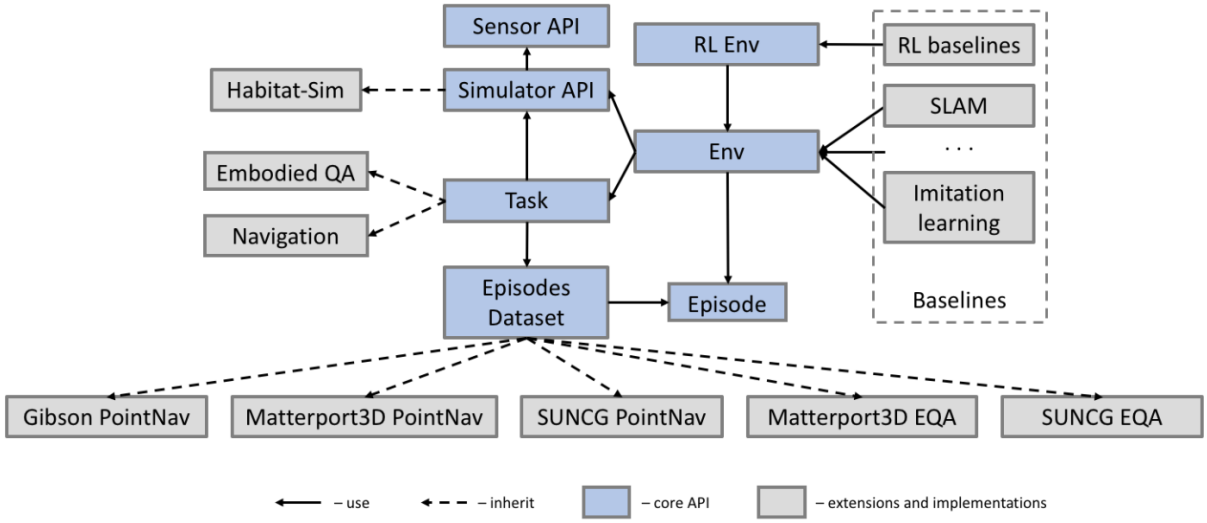
方法

本作業利用到 Habitat-API、Python、CUDA、Pytorch、PPO、Split+PPO。以下將介紹 Habitat-API、論文(註 1)提供的 Baseline、PPO、Split+PPO。

一、Habitat-API

模擬 3D 環境。

Python 藉由呼叫 Habitat-API，獲得環境資訊(影像、深度、GPS、方位)，利用演算法(RL)訓練 Agent 達成任務。



Env

模擬 3D 環境，本作業使用 Gibson 資料集(註 2)，88 種場景。Agent 可以在環境中移動。

(示意圖)

Task
在 Env 中可訓練、評估的任務，包含:導航(Navigation)、問題詢問(Embodied QA)、探索。
定義任務的目的、位置、成功條件、時間限制…。

Sensor API
可以藉由 Sensor 獲得 Env 中的數值，包含:RGB 影像、深度影像(depth)、GPS、羅盤(目標方位)。
要使用哪些 Sensor 資料，取決於演算法要哪些資料，可以只使用 RGB 訓練，也可以只使用 depth 訓練…。

Agent
代理人。控制機器人在 Env 中走動，包含動作(step): 停(任務結束)、前進(0.25 公尺) 、左轉(10 度)、右轉(10 度)。

Episode
Agent 執行一次 Task 的完整過程。Agent 每一個 frame 都可以獲得 Sensor 資訊，藉由獲得的資訊，Agent 做下一步的動作(step)。當超過 500 frame 的動作後，此 Episode 自動結束，任務若沒完成，即算失敗。

二、Baselines
Habitat(註 1) 論文用導航來當作 Baselines 的目標任務。以各類演算法來執行導航任務，給予使用者一個評斷演算法的基準(圖 2)。

導航 (Navigation)
在 Env 中，Agent 在一開始可以藉由 GPS，獲得目的地位置以及自身位置。
此任務目的:從自身位置走到目的地位置。
以下影片範例 (影片 1) ，為由上而下的俯視圖，用來觀測結果的好壞。

episode=873-ckpt=1-distance_to_g

(影片 1)

Success weighted by Path Length (SPI) (註 4)
Habitat 採用此評量準則(圖 1)，以下為評量準則的公式。SPI 越高代表演算法結果越好。

$$\frac{1}{N}\sum_{i=1}^{N} S_i \frac{\ell_i}{\max(p_i, \ell_i)}$$

(圖 1)

Si ： 此次 Episode 的任務是否成功，成功為 1，失敗為 0
Pi ： Agent 在此次 Episode 共走了多少距離
li ： 最短路徑
i ： Episode 第 i 個

Reward function (圖 2)

Habitat-API 會根據執行的任務，回饋 Agent 剛剛所執行動作的好壞。以下為導航任務的回饋數值。

$$r_t = \begin{cases} s + d_{t-1} - d_t + \lambda & \text{if goal is reached} \\ d_{t-1} - d_t + \lambda & \text{otherwise} \end{cases}$$

(圖 2)

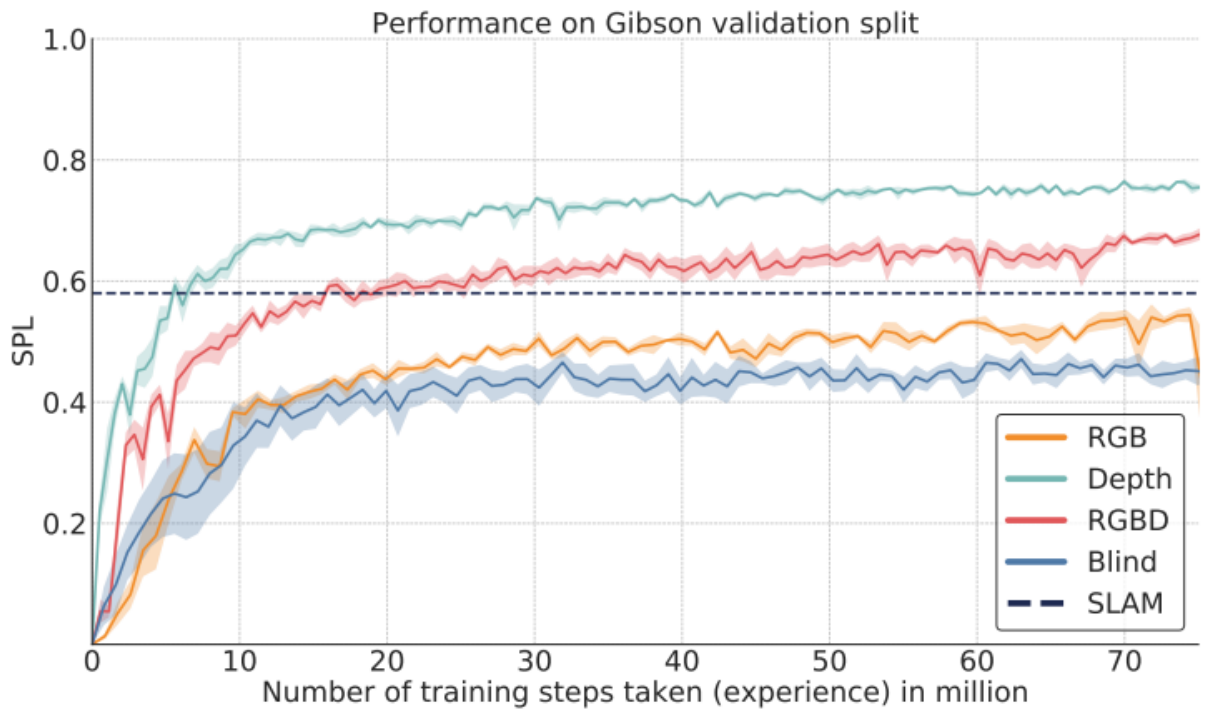
s ： 若成功的獎勵 (10)
λ ： 時間逞罰 (-0.01)
dt ： 當前離目的地的距離
dt-1 ： 上一幀離目的地的距離

演算法

比較五種算法結果(圖 3)，分別訓練 7 千 5 百萬次 step。
只有 Depth 的 RL 結果最好，論文推測有 RGB 的訓練結果會因為 overfitting 效能下降。

1. RL (PPO)
   - 甲、RGB
     - i. 使用 RGB
     - ii. 訓練時間 566 小時
   - 乙、Depth
     - i. 使用 Depth
     - ii. 訓練時間 475 小時
   - 丙、RGBD
     - i. 使用 RGB 和 Depth
     - ii. 訓練時間 906 小時
   - 丁、Blind
     - i. 完全不使用影像
     - ii. 訓練時間 320 小時

2. SLAM(The simultaneous localization and mapping)(註 5)
   - 甲、SLAM 不需要訓練

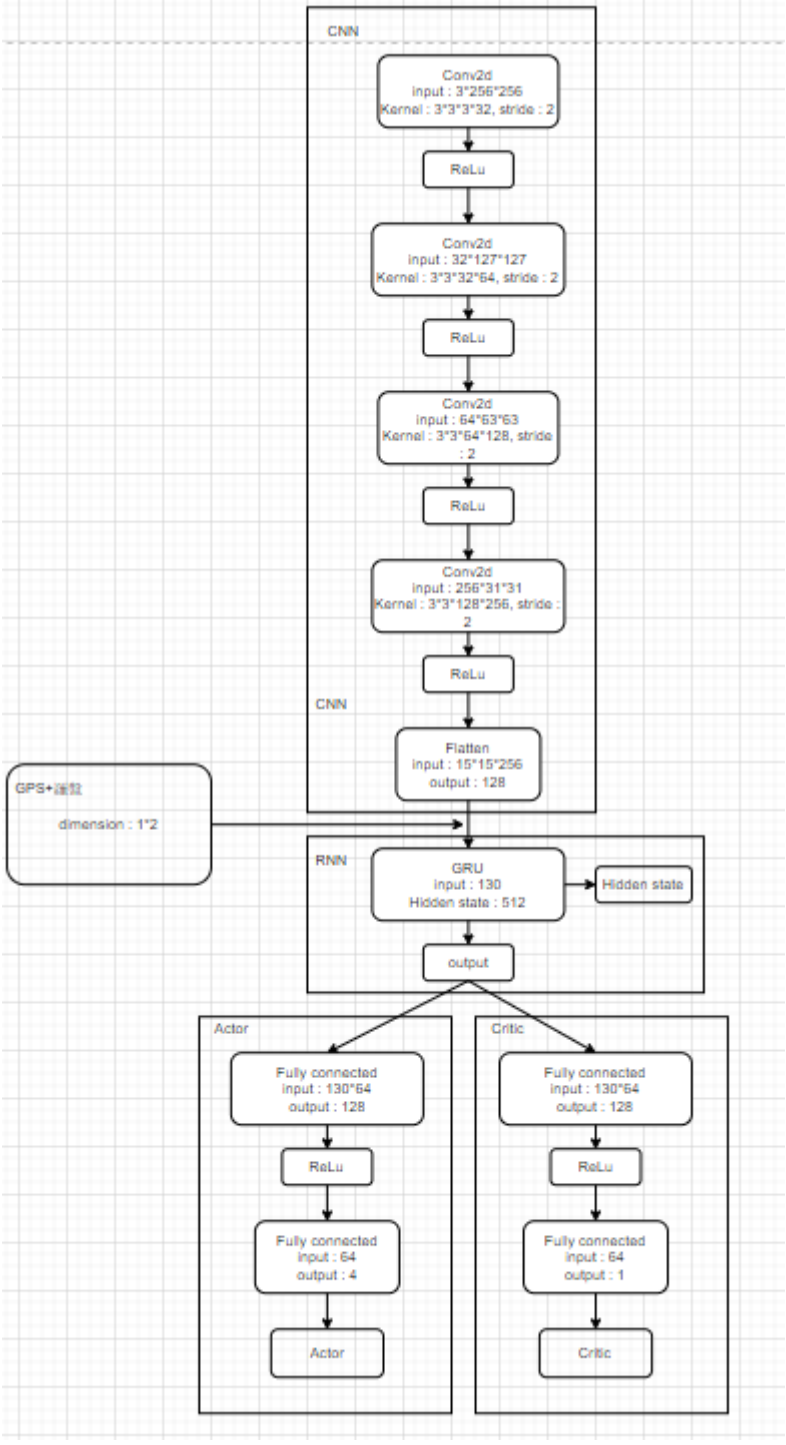


(圖 3)

三、PPO (Proximal Policy Optimization Algorithms)(註 6)

PPO 演算法為強化學習的其中一種演算法。強化學習藉由獲得每一幀的 depth image、GPS(當前位置)、羅盤(目的地方向)，決定 Agent 該執行哪個動作(step)。

本作業粗略實作 PPO 演算法，以下為網路架構。

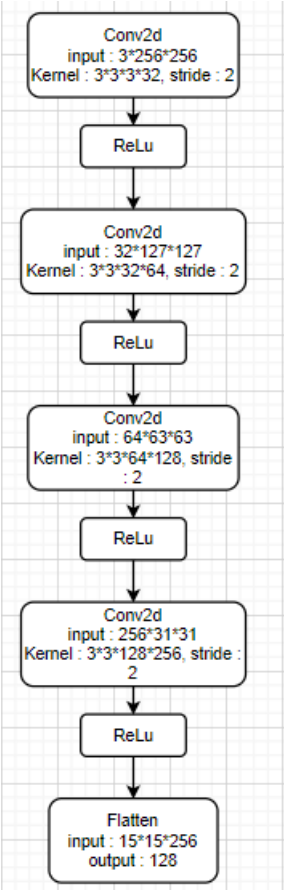


Replay Buffer

紀錄前數次 Episode 的所有資料(每一步 step)，包含 Depth image、Reward、RNN 的 Hidden state、Actor 輸出、Critic 輸出。用來之後倒傳遞訓練網路。

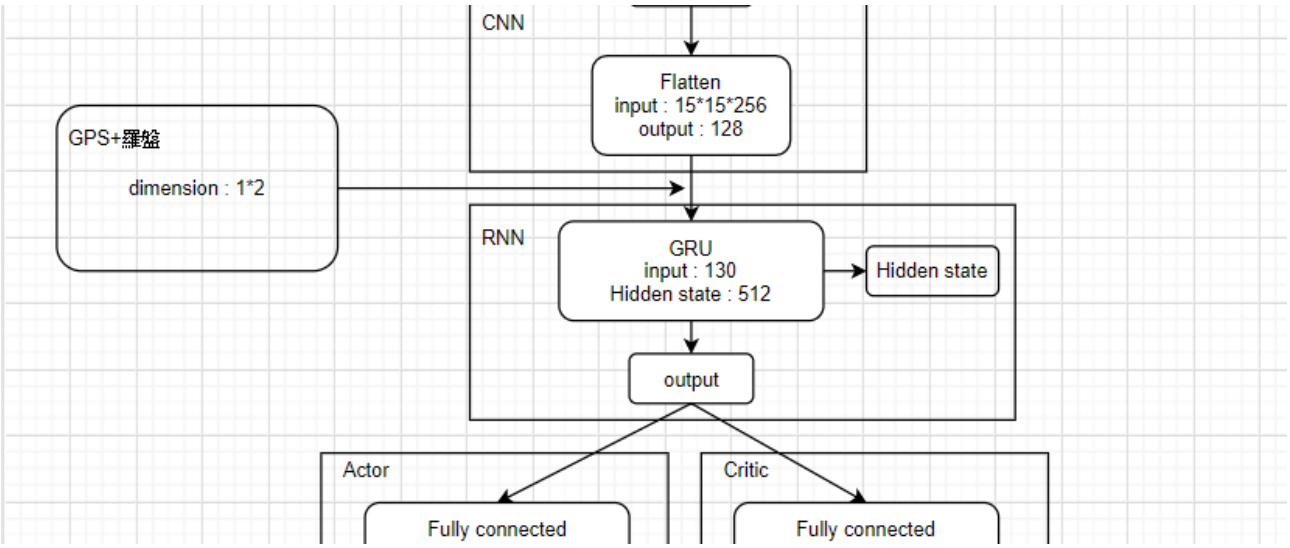
CNN Encoder

將 depth image 當作 input，丟入連續的捲積層當中，獲得較低維特徵。



RNN

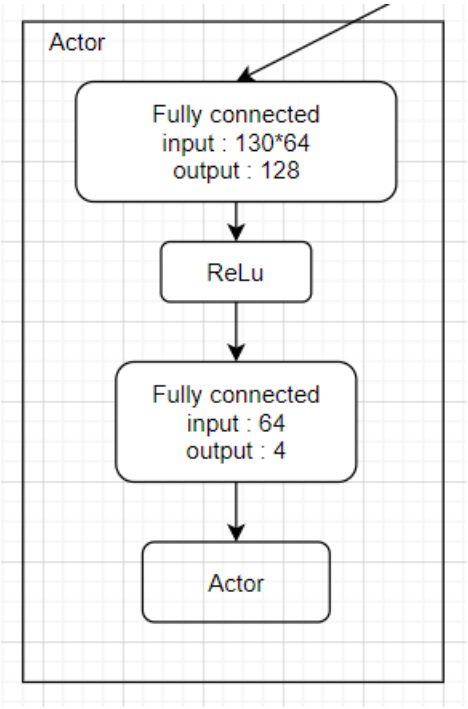
將 CNN 獲得的特徵+GPS+羅盤數值，丟入 RNN 當中，輸出給 Actor 和 Critic 使用。



Actor

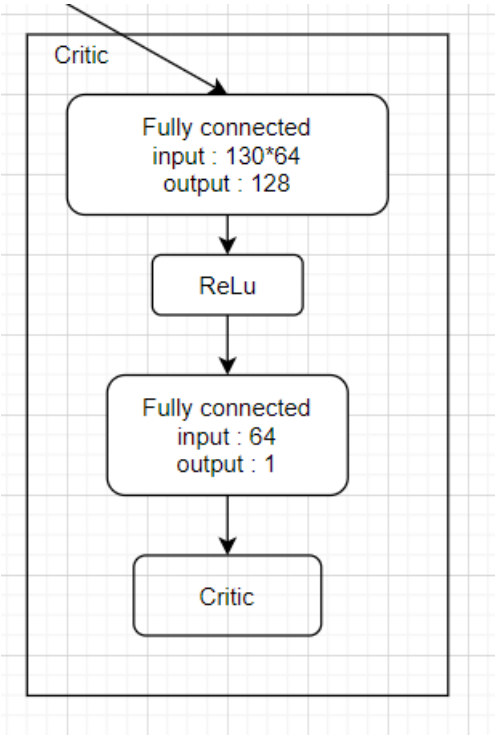
Agent 根據輸出，選擇 4 個動作 (step)的其中之一。

包含動作 ： 停(任務結束)、前進(0.25 公尺) 、左轉(10 度)、右轉(10 度)。



Critic

評論當前 depth image 能夠獲得的未來獎勵(reward)。



PPO 演算法目標函數

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right]$$

$L_t^{CLIP}(\theta)$: actions loss

透過累加每一幀動作(step)獲得的 reward，來評估此神經網路的好壞。Reward 總合越高，代表此網路越好，因此此項 loss 越高越好。

$L_t^{VF}(\theta)$: values loss

Critic 網路輸出數值，預測 reward，當 reward 預測越準確，loss 越小。

$S[\pi_\theta](s_t)$: distributions entropy

Actor 4 種動作的機率越平均，entropy 越高，每個動作就都有機會被採取，越有機會探索到最佳策略。

結果

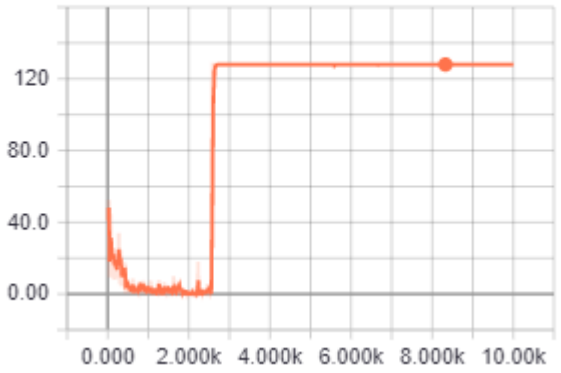
訓練 10000 次 epochs，下圖（圖 4）為 PPO 演算法的參數設置。

```
RL:
  PPO:
    # ppo params
    clip_param: 0.1
    ppo_epoch: 4
    num_mini_batch: 1
    value_loss_coef: 0.5
    entropy_coef: 0.01
    lr: 2.5e-4
    eps: 1e-5
    max_grad_norm: 0.5
    num_steps: 128
    cnn_output_size: 128
    hidden_size: 512
    use_gae: True
    gamma: 0.99
    tau: 0.95
    use_linear_clip_decay: True
    use_linear_lr_decay: True
    reward_window_size: 50

    use_splitnet_auxiliary: False
```

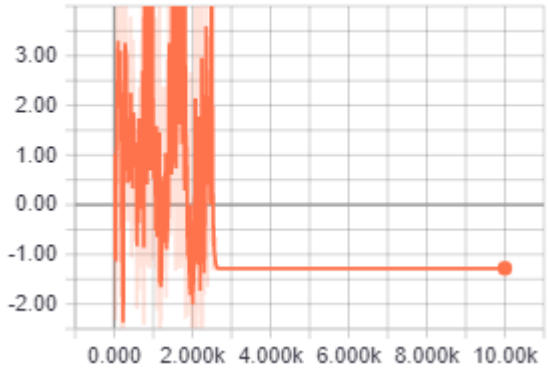
(圖 4)

此演算法剛開始時，能夠逐漸收斂優化。但後來 Agent 僅在 Task 剛開始時，就直接採取”停”的策略，Agent 無法學習導航技能。結果極差。SPI 最高為 35%，成功導航 45%。

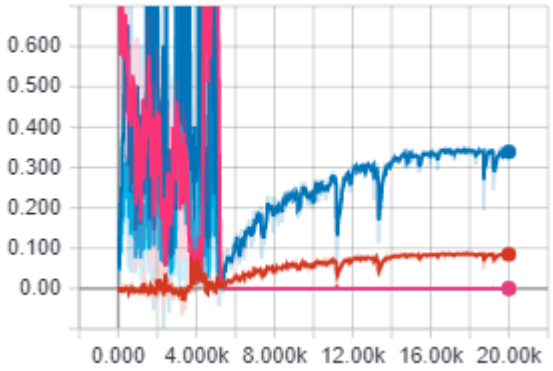
count



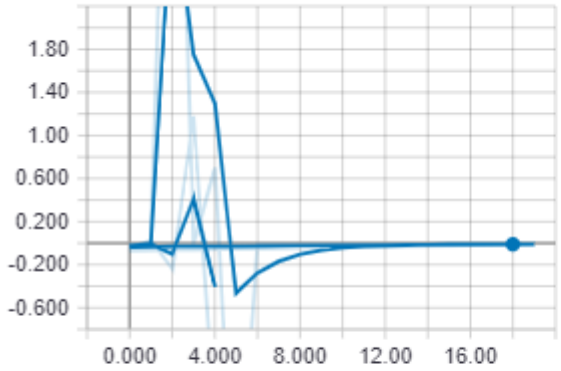
reward



loss



eval_reward



Name	Smoothed	Value	Step	Time	Relative
eval_reward_average reward	-0.01057	-1.0000e-2	18.00	Fri May 29, 22:10:05	6m 27s

以下影片（影片 2)(影片 3）為 Agent 尚未收斂時的導航情況。

episode=248-ckpt=4-distance_to_g

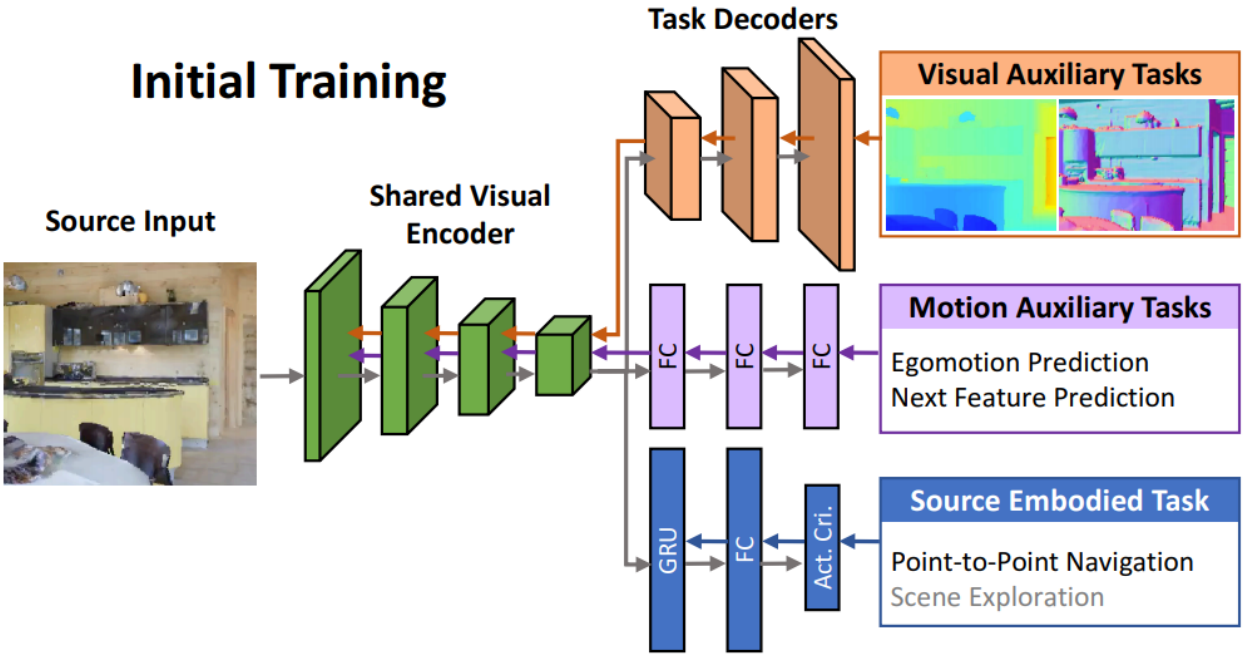
（影片 2）

episode=798-ckpt=6-distance_to_g

（影片 3）

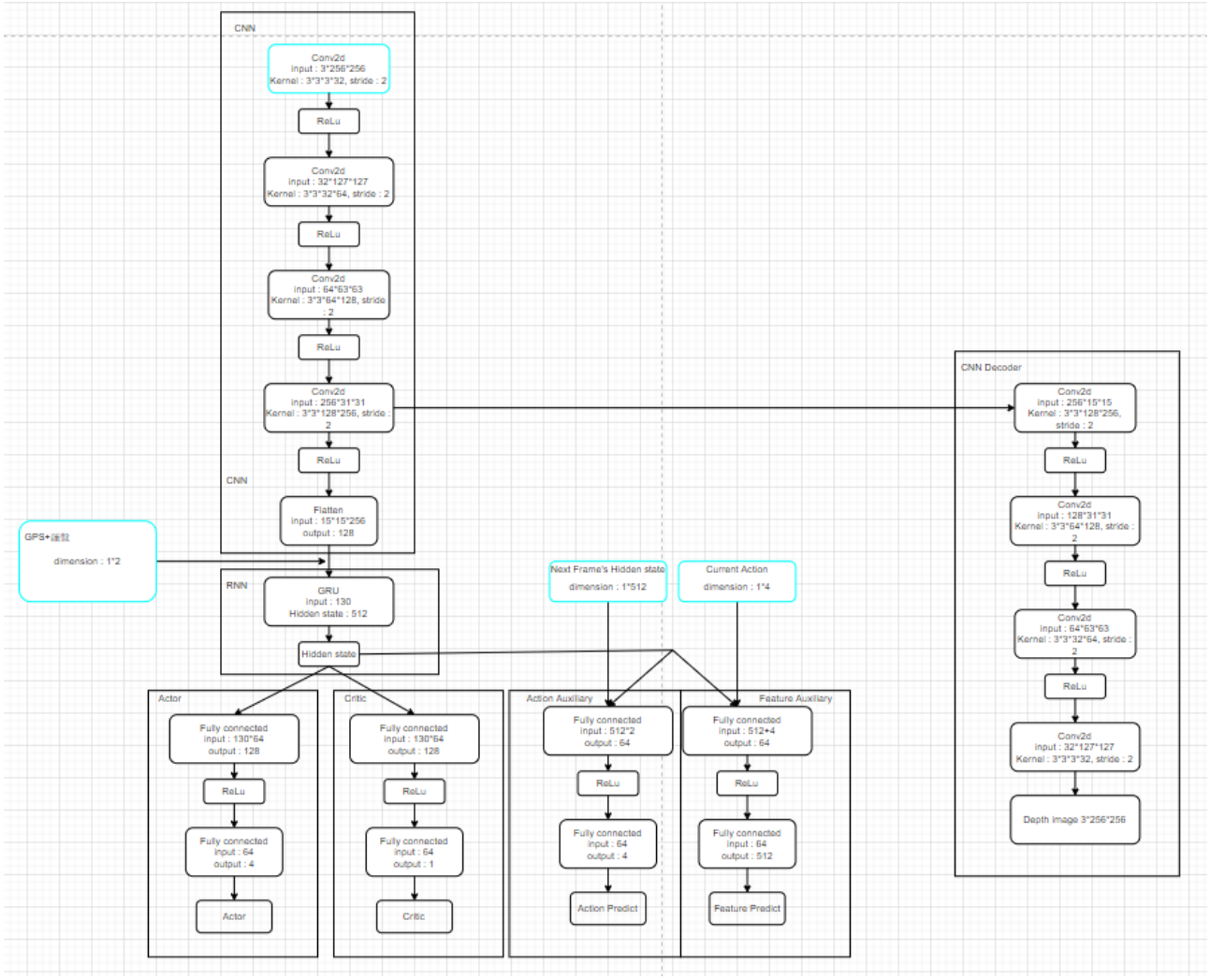
四、SplitNet: Sim2Sim and Task2Task Transfer for Embodied Visual Navigation (註 7)

此論文實驗在導航任務中，除了原本利用 PPO 演算法(Source Embodied Task)、實驗不同任務的 transfer 能力，額外添加輔助網路(Motion Auxiliary Tasks)、影像深度網路(Visual Auxiliary Tasks)，主要是讓 CNN 編碼(Shared Visual Encoder)提取正確特徵，在實際導航過程中，不需要 Auxiliary 的輸出。



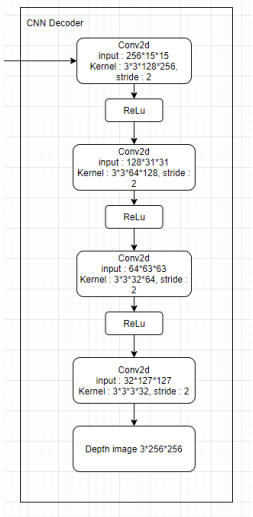
（註 7）

本作業粗略實作 SplitNet 方法，結合 三、PPO 的網路架構，稍微改變輸入資料、網路架構。以下為網路架構。



CNN Decoder

將 CNN Encoder 所獲得的低解析度特徵圖，進行上採樣，獲得 3*256*256 的影像，跟原本 CNN Encoder 輸入影像，做均方誤差更新網路。

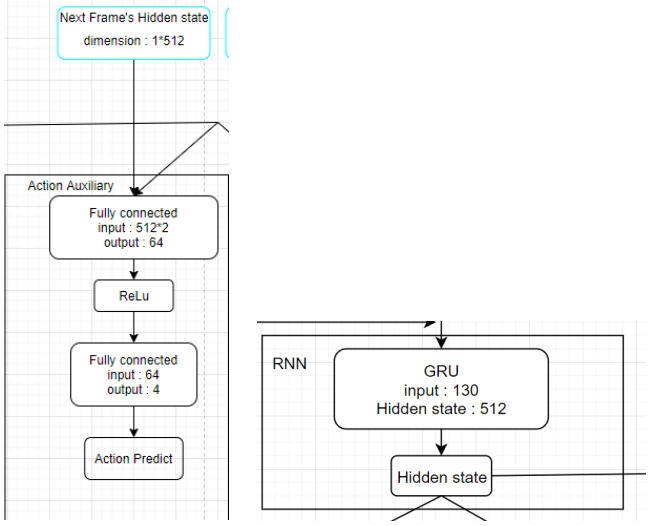


Action Auxiliary

Input ： 當前幀的 RNN Hidden state + 下一幀的 RNN Hidden state

Output ： 動作(step)

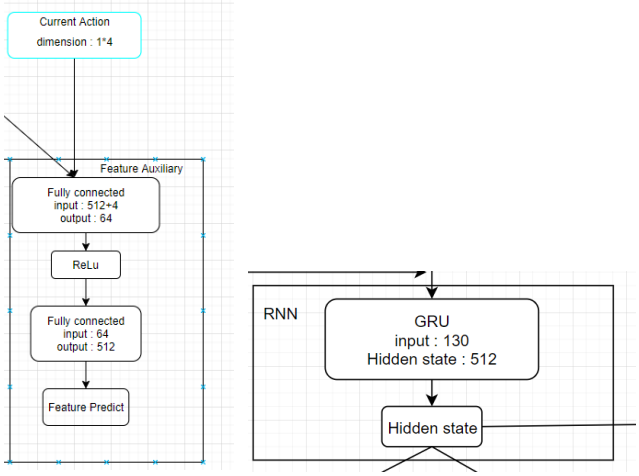
利用”當前幀的 RNN Hidden state”和“下一幀的 RNN Hidden state”，期望能夠預測出是哪一個動作(step)，導致從”當前幀的 RNN Hidden state”改變成“下一幀的 RNN Hidden state”。



Feature Auxiliary

Input ： 當前幀的 RNN Hidden state + 當前 Actor 採取的動作(step)
Output ： 下一幀的 RNN Hidden state

利用” 當前幀的 RNN Hidden state”和 ” 當前 Actor 採取的動作(step)” ，預測” 下一幀的 RNN Hidden state”。



SplitNet Auxiliary 目標函數

$$\mathcal{L}_D = \sum_{pixels} \|\mathcal{D}(\phi_t) - D_t\|_1$$ ： CNN Decoder loss

真實深度圖跟解碼出來的深度圖，彼此之間差距越小越好。

$$\mathcal{L}_E = -\sum_{a \in A} p(a_t = a) \log(\mathcal{E}(\phi_t, \phi_{t-1}))$$ ： Action Auxiliary loss

利用 cross entropy 計算損失函數，比對 Ground True Action 和網路預測的 Action 差距，差距越小越好。

$$\mathcal{L}_P = 1 - \sum_{features} \frac{\mathcal{P}(\phi_{t-1}, a_t) \cdot \phi_t}{\|\mathcal{P}(\phi_{t-1}, a_t)\|_2 * \|\phi_t\|_2}$$: Feature Auxiliary loss

利用”預測的下一幀 Hidden state”以及”真實下一幀 Hidden state”，使用 cosine 比對兩者相似度，最後被 1 減，就是 Feature Auxiliary 的 loss

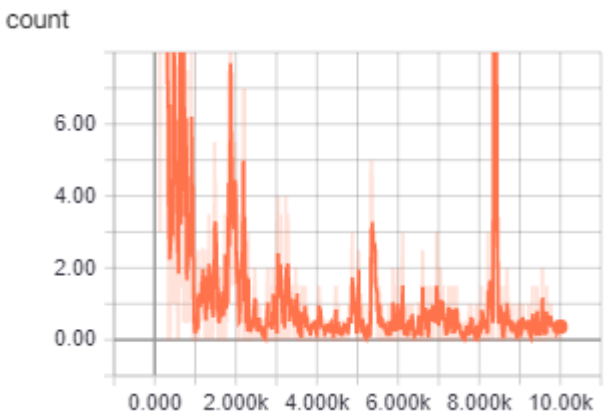
結果

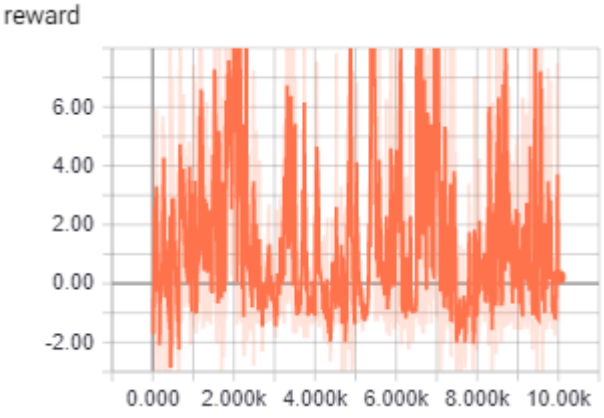
訓練 10000 次 epochs，下圖（圖 5）為 Split+PPO 演算法的參數設置。

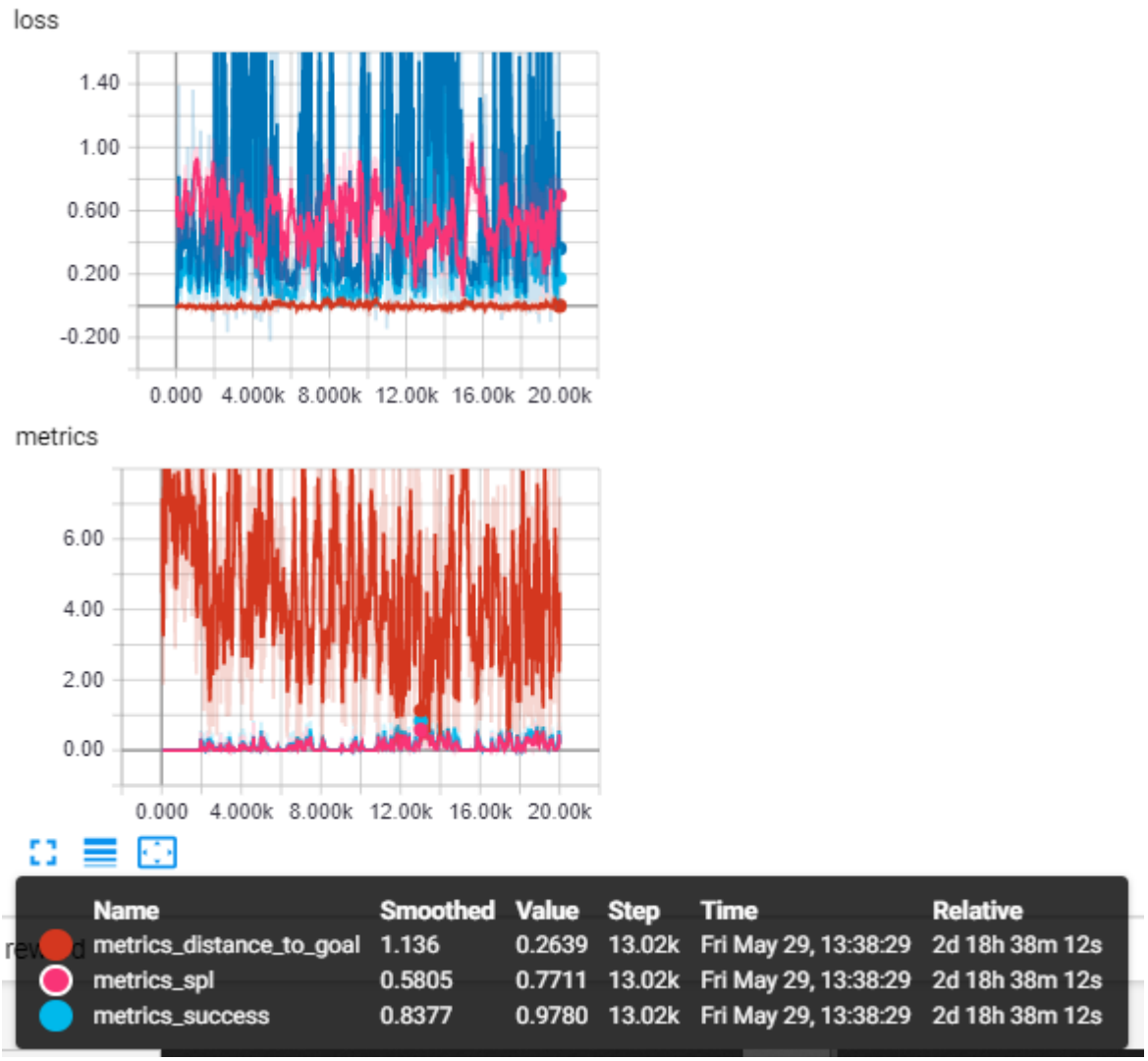
```
RL:
  PPO:
    # ppo params
    clip_param: 0.1
    ppo_epoch: 4
    num_mini_batch: 1
    value_loss_coef: 0.5
    entropy_coef: 0.01
    lr: 2.5e-4
    eps: 1e-5
    max_grad_norm: 0.5
    num_steps: 128
    cnn_output_size: 128
    hidden_size: 512
    use_gae: True
    gamma: 0.99
    tau: 0.95
    use_linear_clip_decay: True
    use_linear_lr_decay: True
    reward_window_size: 50

    use_splitnet_auxiliary: True
```

（圖 5）

相較於之前的 PPO 結果，Split+PPO 演算法能夠預防 Agent 最終直接採取”停”的策略。SPI 可以到達最高 0.58 的結果，成功導航 83%。比起 PPO 演算法更加穩定。







以下影片（影片 4)(影片 5）為 Agent 導航成功以及導航失敗的影片。

episode=693-ckpt=8-distance_to_g

（影片 4）

episode=970-ckpt=9-distance_to_g

（影片 5）

參考資料

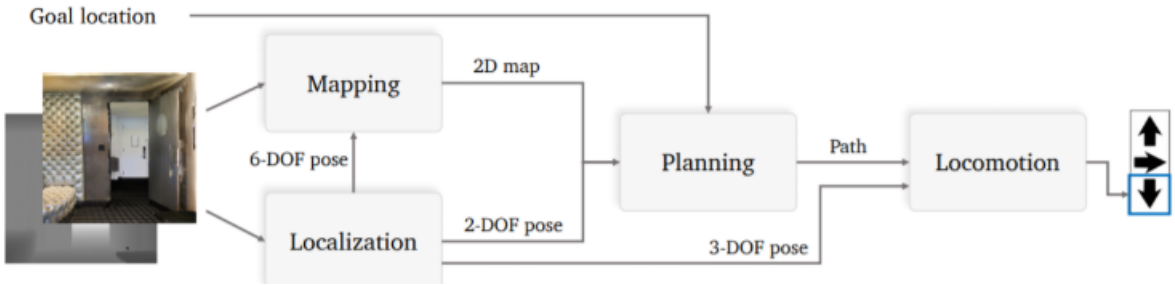
[1] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, Dhruv Batra. Habitat: A Platform for Embodied AI Research. arXiv:1904.01201, 2019

[2] Gibson (http://gibsonenv.stanford.edu/database/)
3D 環境模型

[3] Habitat-API (https://github.com/facebookresearch/habitat-api)

[4] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, Amir R. Zamir. On Evaluation of Embodied Navigation Agents. arXiv:1807.06757, 2018

[5] Dmytro Mishkin, Alexey Dosovitskiy, and Vladlen Koltun. Benchmarking classic and learned navigation in complex 3D environments. arXiv:1901.10915, 2019.



拆解成 4 個小模組分工

- mapping
  - outputs a two-dimensional obstacle map of the environment
  - 將深度影像投影到 local map；之後合併到 Global map；做路徑規劃。
- localization
  - estimates the agent's pose
- planning
  - plan a path to the goal
- locomotion
  - selects an action to follow the path

[6] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov. Proximal Policy Optimization Algorithms. arXiv:1707.06347, 2017

[7] Daniel Gordon, Abhishek Kadian, Devi Parikh, Judy Hoffman, Dhruv Batra. SplitNet: Sim2Sim and Task2Task Transfer for Embodied Visual Navigation. arXiv:1905.07512, 2019