Cartpole

Overall Architecture

Agent (abstract)
- RandomAgent
- QLearningCartPoleAgent

StateDiscretizer

Trainer

- Agent：定義行為介面
- RandomAgent / QLearningAgent：繼承並 override
- StateDiscretizer：負責觀測 → 離散狀態
- Trainer：統一管理訓練與測試流程

Class: Agent (Abstract Base Class)

方法	用途
begin_episode()	切換訓練/測試模式
select_action()	由子類別決定策略（抽象）
learn()	子類別決定如何學習（抽象）
end_episode()	可 override 的 per-episode 操作

select_action() 和 learn() 是 abstractmethod → 子類別必須 override
→ 多型（polymorphism）。

Class: RandomAgent

用於展示「繼承 + overriding」的最小例子

- 繼承自 Agent
- select_action() 覆寫成純隨機
- learn() override 成空實作

不同 Agent 共享共同介面（多型），Trainer 不需修改任何程式就可以使用。

Class: StateDiscretizer

封裝（Encapsulation）示範

功能：

- CartPole observation 是連續值
- 離散化是 Q-table 需要的
- 處理 clipping、normalization、binning

離散化邏輯和 Agent 分開 → 好維護 → QLearningAgent 不需知道這些細節。

Class: QLearningCartPoleAgent

- Override select_action
  （epsilon-greedy）
- Override learn（Q-learning 更新）
- Override end_episode（epsilon decay）

  - QLearningCartPoleAgent 是 一個策略（Policy） 的 封裝
  - 可以隨時被 Trainer 替換

Q-Learning 更新公式

在 learn() 中使用：

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma a' \max Q(s',a') - Q(s,a))$$

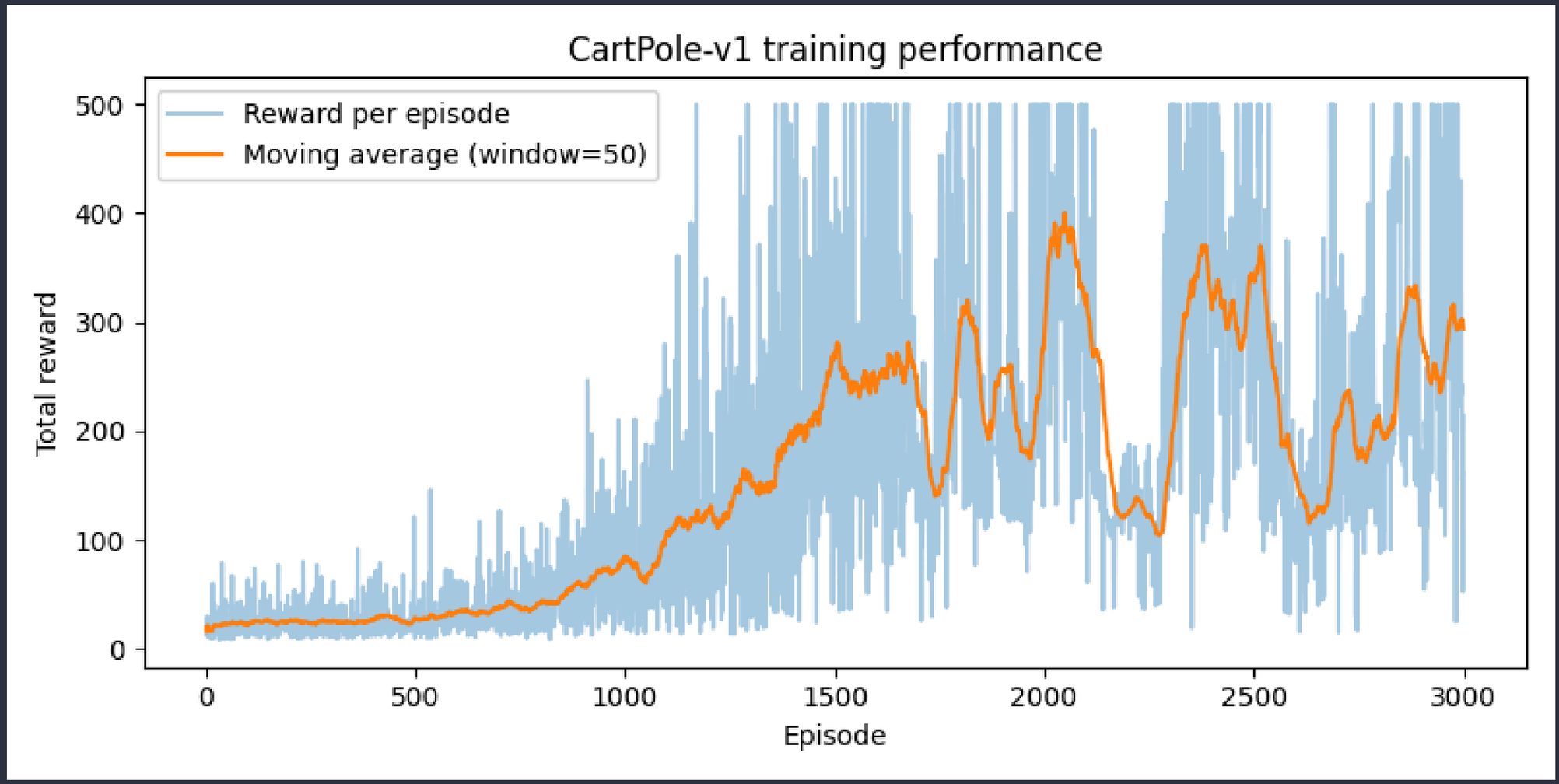
Class: Trainer

高抽象度的架構設計：

功能：

- 控制 episode loop
- 呼叫 agent 的：
  - select_action()
  - learn()
  - begin_episode() / end_episode()
- 訓練跟測試用同一套流程
- 不需知道 agent 內部如何實作 → 這就是 多型（polymorphism）

Training Curve



Demo Video

Summary

Encapsulation：

Agent、Discretizer、Trainer 職責分明

每個 class 各自封裝自己的邏輯

Inheritance：

RandomAgent、QLearningAgent 都繼承自 Agent

Polymorphism：

Trainer 不需知道是哪個 Agent

只要呼叫 agent.select_action() 就能運作

Abstraction：

基底 Agent/Discretizer 抽象化界面，不處理細節

Extensibility：

換成 DQN、SARSA、Actor-Critic 只要寫新 Agent class 就能無痛加入