

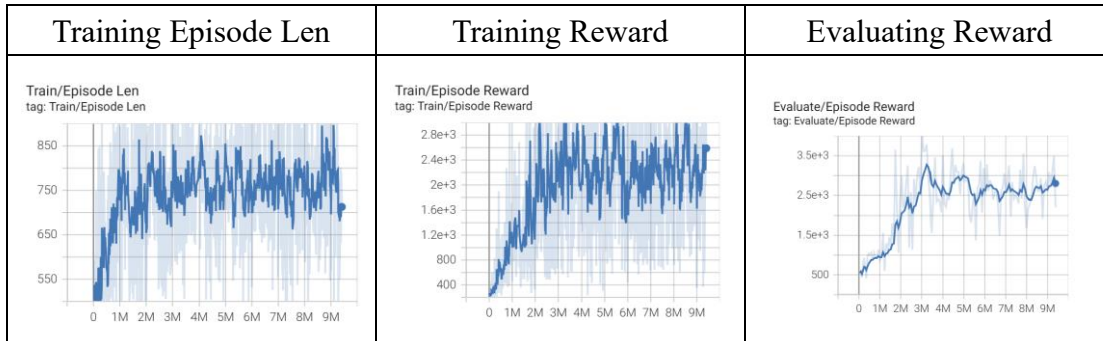
# RL Lab 2 Report

資科工碩 陳冠廷 313551058

## ● Basic

### ■ Screenshot of Tensorboard training curve and testing results on DQN.

Training Curve of DQN on game MsPacman-v5



Testing results of DQN on game MsPacman-v5

```
=====
Evaluating ...
episode 1 reward: 4260.0
episode 2 reward: 2710.0
episode 3 reward: 2350.0
episode 4 reward: 3690.0
episode 5 reward: 3810.0
average score: 3364.0
=====
```

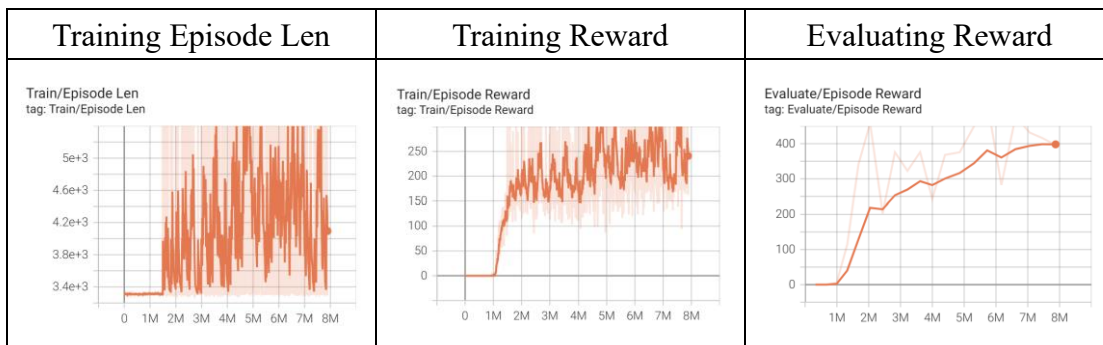
**PS:** 本次實驗的內容（MsPacman-v5 與 Enduro-v5）都是用相同的參數設定，參數值依照原先的 template 設定，僅將 batch\_size 改為 64。由於實驗要跑很久，當程式執行到還不錯的結果時，我會手動中斷程式，因此實驗結果的 training\_step 與設定值不同。所有截圖都是將 tensorboard 的 smooth 設定成 0.8 的繪圖結果。

Observation 我會將其 resize 至 (84, 84) 並且以 grayscale 的讀取，用以減少參數量外，也可以保留有用的資訊即可。同時，為了讓每個輸入更有意義，我用 framestack 的方式堆疊多個 frame 讓每個輸入可以看到更有意義的內容。例如：遊戲裡的物件以單個 frame 來看沒辦法捕捉到方向和速度的概念，但是當堆疊多個 frame 後就可以有時間上的差異，進而較有機會捕捉到速度與移動方向的概念。

## ● Bonus

### ■ Screenshot of Tensorboard training curve and testing results on Enduro-v5 using DQN.

## Training curve of DQN on game Enduro-v5



## Testing results of DQN on game Enduro-v5

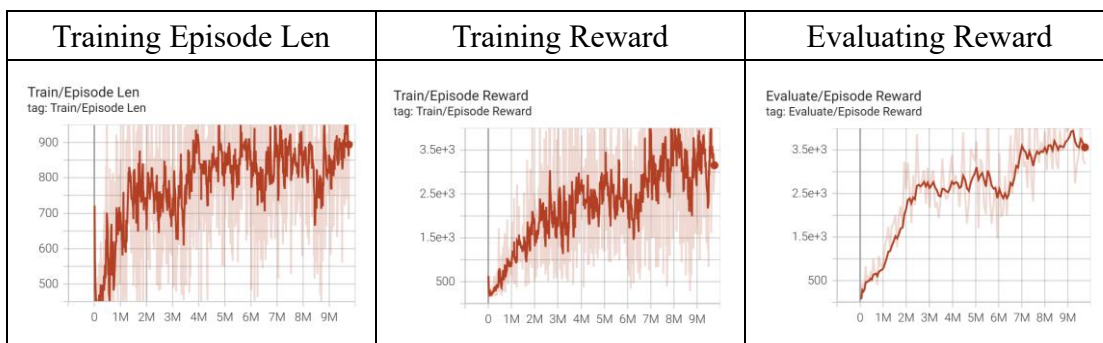
```
=====
Evaluating ...
episode 1 reward: 483.0
episode 2 reward: 481.0
episode 3 reward: 441.0
episode 4 reward: 650.0
episode 5 reward: 699.0
average score: 550.8
=====
```

**PS:** Enduro-v5 訓練過程會有一些特性：前面 1M 訓練過程的 reward 都是 0 的狀況，最後突然往上升，目前推測因為這個遊戲有很多 action 是多餘的，大多時候左右應該就很夠用了，而且他也不像小精靈可以吃分數馬上就有 reward 可以得到分數，而是需要超過車才會得到 reward，因此前期需要嘗試多次才知道那些動作較沒用，並且嘗試多次後才可以超過車成功獲得一些 reward。

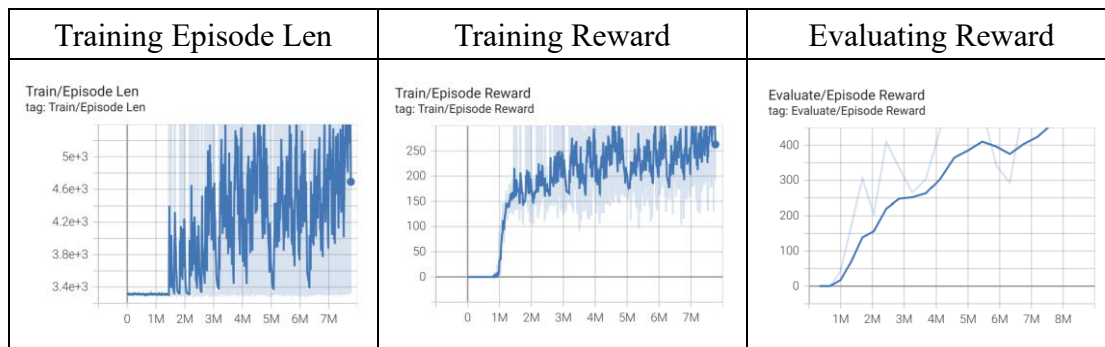
## ■ Screenshot of Tensorboard training curve and testing results on DDQN, and discuss the difference between DQN and DDQN.

### Training curve of DDQN on 2 games

#### ◆ MsPacman-v5



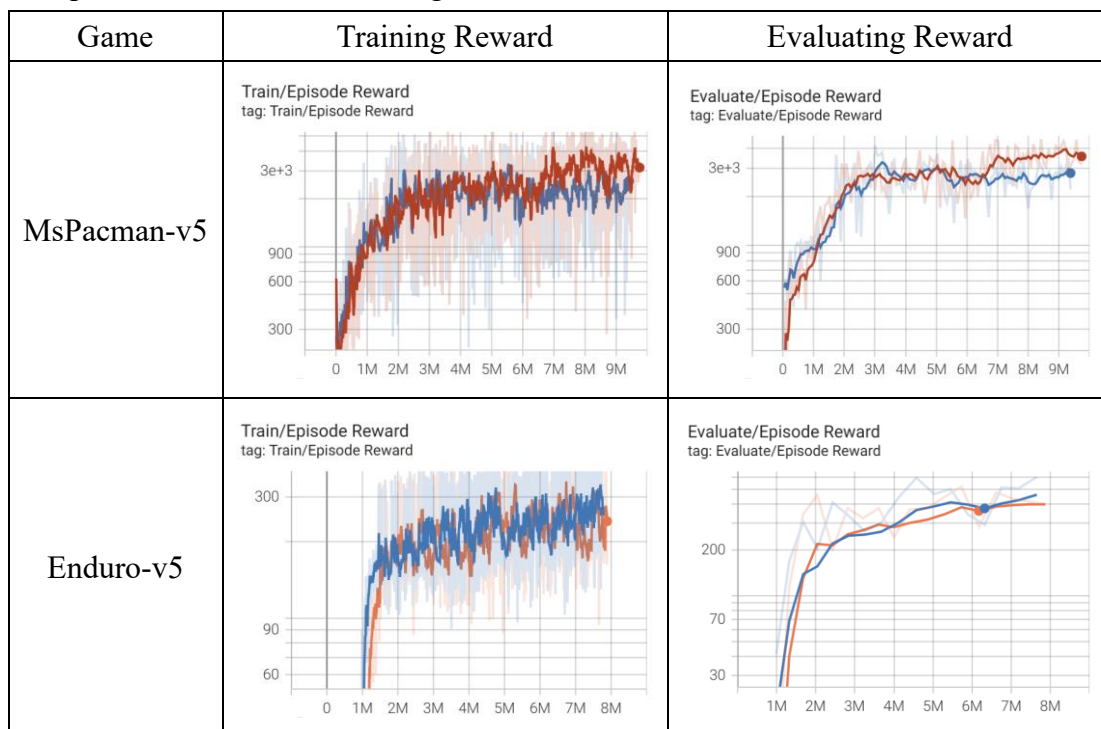
## ◆ Enduro-v5



## Testing results of DDQN on 2 games

MsPacman-v5	Enduro-v5
<pre>===== Evaluating ... episode 1 reward: 3990.0 episode 2 reward: 4160.0 episode 3 reward: 4430.0 episode 4 reward: 4230.0 episode 5 reward: 3860.0 average score: 4134.0 =====</pre>	<pre>===== Evaluating ... episode 1 reward: 486.0 episode 2 reward: 469.0 episode 3 reward: 726.0 episode 4 reward: 486.0 episode 5 reward: 736.0 average score: 580.6 =====</pre>

## Compare DQN and DDQN on 2 games



## Discussion:

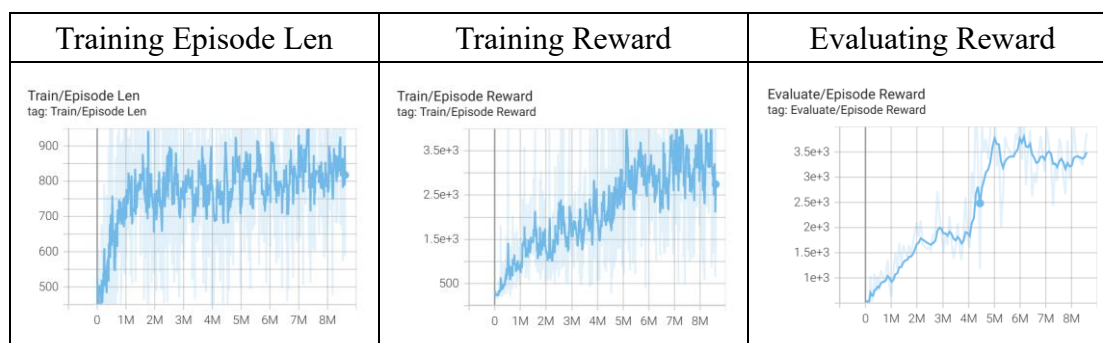
DDQN 與 DQN 在推理的過程沒有差異，都是選擇 Q 值最大（behavior network 的輸出）的動作來執行，並根據一定的機率 epsilon 來隨機探索。它們唯一的差距就在於訓練時的目標函數，DDQN 認為 DQN 容易有 over-estimate 的問題，因此改進了預期 Q 值的計算方法。DQN 直接用 target network 的最大 Q 值當作 TD Learning 的目標，而 DDQN 則是先用 behavior network 來找出最大 Q 值的動作，並利用 target network 算出該動作的 Q 值，當

作 TD learning 的目標。因為原始 DQN 不論動作或是 Q 值的選擇都是透過 target network 容易有過度樂觀的問題，容易有偏差的問題。從結果也可以看到 DDQN 的結果通常略好於 DQN，而且震盪幅度較小(DDQN 在 MsPacman 為紅色的曲線，Enduro 為藍色的)。

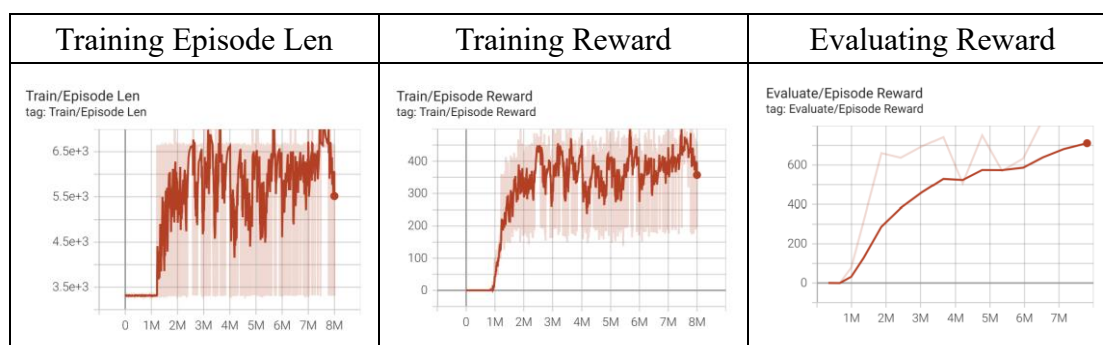
## ■ Screenshot of Tensorboard training curve and testing results on Dueling DQN, and discuss the difference between DQN and Dueling DQN.

Training curve of Dueling DQN on 2 games

### ◆ MsPacman-v5



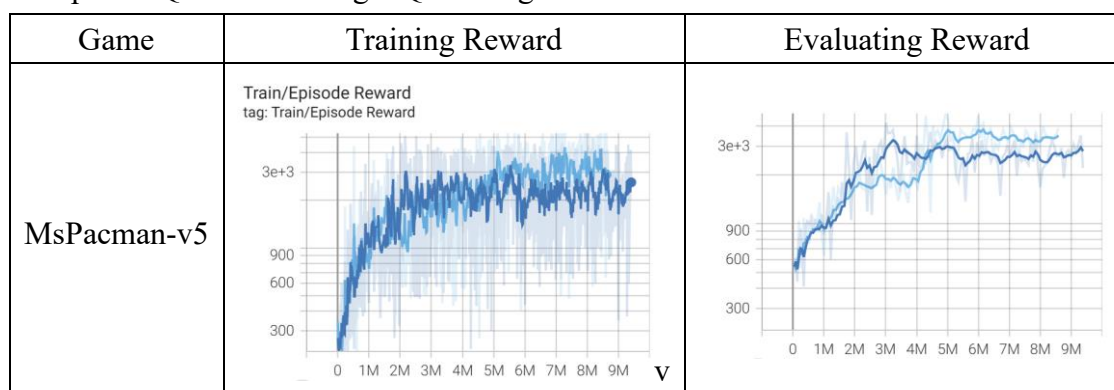
### ◆ Enduro-v5

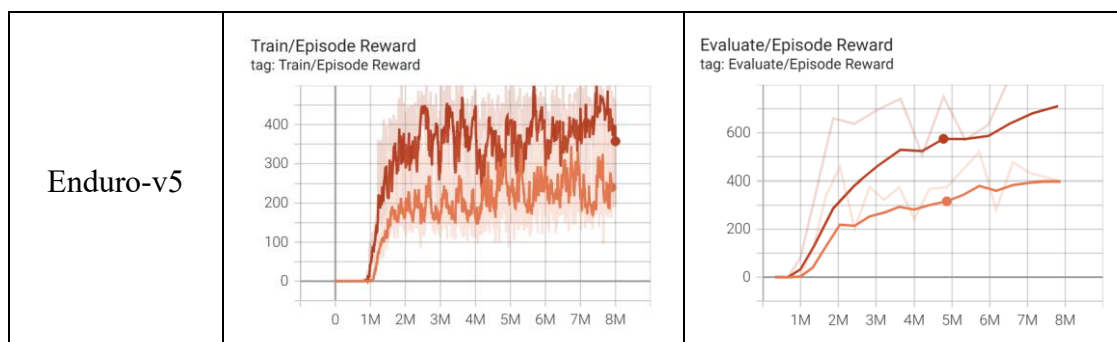


Testing results of Dueling DQN on 2 games

MsPacman-v5	Enduro-v5
<pre> ===== Evaluating ... episode 1 reward: 4850.0 episode 2 reward: 4160.0 episode 3 reward: 5360.0 episode 4 reward: 2460.0 episode 5 reward: 5370.0 average score: 4440.0 ===== </pre>	<pre> ===== Evaluating ... episode 1 reward: 760.0 episode 2 reward: 782.0 episode 3 reward: 1049.0 episode 4 reward: 1054.0 episode 5 reward: 1032.0 average score: 935.4 ===== </pre>

Compare DQN and Dueling DQN on 2 games





## Discussion:

Dueling DQN 與 DQN 在推理的過程沒有差異，都是選擇 Q 值最大（behavior network 的輸出）的動作來執行，並根據一定的機率 epsilon 來隨機探索。它們最大的差異在於模型的架構和 Q 值的計算方法，Dueling DQN 分別計算了 Value 與 Advantage 兩個數值，並根據以下方式來計算 Q 值：

```
class AtariNetDuelDQN(nn.Module):
    def __init__(self, num_classes=4, init_weights=True):
        self.cnn = nn.Sequential(nn.Conv2d(4, 32, kernel_size=8, stride=4),
                                nn.ReLU(True),
                                nn.Conv2d(32, 64, kernel_size=4, stride=2),
                                nn.ReLU(True),
                                nn.Conv2d(64, 64, kernel_size=3, stride=1),
                                nn.ReLU(True)
                                )

        # predict value
        self.value = nn.Sequential(nn.Linear(7*7*64, 512),
                                   nn.ReLU(True),
                                   nn.Linear(512, 1)
                                   )

        # Advantage stream
        self.advantage = nn.Sequential(nn.Linear(7*7*64, 512),
                                       nn.ReLU(True),
                                       nn.Linear(512, num_classes)
                                       )

        if init_weights:
            self._initialize_weights()

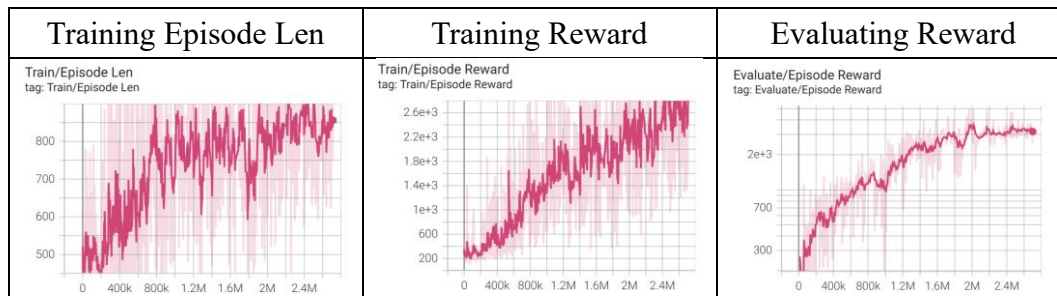
    def forward(self, x):
        x = x.float() / 255.
        x = self.cnn(x)
        x = torch.flatten(x, start_dim=1)
        value = self.value(x)
        advantage = self.advantage(x)
        return value + (advantage - advantage.mean(dim=1, keepdim=True))
```

模型透過 CNN 來學習特徵並共享，利用兩個不一樣的 Linear 層分別預測 value 和 advantage。本次實驗的訓練方式是採用 DDQN 的不過度樂觀的訓練方法，因此從實驗結果可以看到 Dueling DQN 都比 DQN 好很多，而且特別是在 Enduro 這種需要看到更細節的動作分數的遊戲會有更好的表現，遠勝於 DQN。（因為 advantage 可以更細節地看出每個動作的差異，預測 Advantage 也有助於模型學習，因為 advantage 通常值的差異較小）

- Screenshot of Tensorboard training curve and testing results on DQN with parallelized rollout, and discuss the difference between DQN and DQN with parallelized rollout



## Training curve of Parallelized rollout DQN on MsPacman-v5



## Testing results of Parallelized rollout DQN on MsPacman-v5

```
=====
Evaluating ...
episode 1 reward: 3370.0
episode 2 reward: 3200.0
episode 3 reward: 2460.0
episode 4 reward: 5320.0
episode 5 reward: 6170.0
average score: 4104.0
=====
```

因為時間有限，平行化的環境我只用於 MsPacman-v5，而且執行的時間不多，大約 2.5M 左右。一開始實作時，一直是使用 `AsyncVectorEnv`，但會經常執行到一半報錯並回傳 `NoneType`。因此平行化的環境，我改用 `SyncVectorEnv` 來實作，平行環境與一般環境的差異在於一般環境只能輸入一個 action，而平行化環境可以輸入多個 action，並一次取得多個環境的 feedback，而且平行環境預設下會自動 reset 已經結束的環境，不需要手動重置。(實驗結果為 8 個 env 的平行結果)

◆ 一次初始化多個環境 → `self.envs`

```
class AtariDQNAgentParallel(DQNBaseAgentParallel):
    def __init__(self, config):
        super(AtariDQNAgentParallel, self).__init__(config)
        ### TODO ###
        # initialize env
        self.envs = gym.vector.SyncVectorEnv([
            lambda: self.__make_wrap(config["env_id"])
            for _ in range(self.num_envs)
        ])
```

```
def __make_wrap(self, env_id):
    env = gym.make(env_id)
    env = atari_preprocessing.AtariPreprocessing(env, screen_size=84, grayscale_obs=True, frame_skip=1)
    env = FrameStack(env, 4)
    return env
```

◆ 一次執行多個動作

```
def decide_agent_actions(self, observations, epsilon=0.0, action_space=None):
    """ TODO """
    # get action from behavior net, with epsilon-greedy selection

    if random.random() < epsilon:
        # exploration by randomly pick the action
        actions = np.array([ action_space.sample() for _ in range(len(observations))])
    else:
        observation = torch.FloatTensor(np.array(observations)).to(self.device)
        with torch.no_grad():
            q_value = self.behavior_net(observation)
            actions = q_value.argmax(dim=1).cpu()

    return actions
```

◆ 一次存多個 feedback 到 replay buffer

```
class DQNBaseAgentParallel(ABC):
    def train(self):
        episode_rewards = np.array([ 0.0 for _ in range(self.num_envs) ])
        episode_lens = np.array([ 0.0 for _ in range(self.num_envs) ])

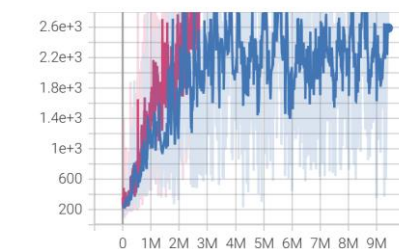
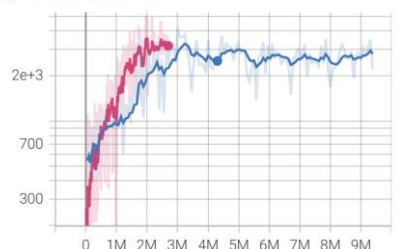
        while self.total_time_step <= self.training_steps:

            if self.total_time_step < self.warmup_steps:
                actions = self.decide_agent_actions(observations, 1.0, self.envs.single_action_space)
            else:
                actions = self.decide_agent_actions(observations, self.epsilon, self.envs.single_action_space)
                self.epsilon_decay()

            next_observations, rewards, terminates, truncates, infos = self.envs.step(actions)

            for i in range(self.num_envs):
                self.replay_buffer.append(
                    observations[i],
                    [actions[i]],
                    [rewards[i]],
                    next_observations[i],
                    [int(terminates[i])])
            )
```

Compare with/without Parallelized rollout

Game	Training Reward	Evaluating Reward
MsPacman-v5	<p>Train/Episode Reward tag: Train/Episode Reward</p> 	<p>Evaluate/Episode Reward tag: Evaluate/Episode Reward</p> 

**Discussion:**

在相同的 TimeStamp 下，可以發現平行化的結果會好過僅有單個 env，因為在相同 TimeStamp 下(2M 的時間點)，可以看到平行化的 evaluate score 已經好過一般的方法一定的數值了，原因就是因為平行化 8 個 environments 意味著，相同時間下平行化已經有 8 倍的經驗量了，所以可以較快執行到很好的分數。