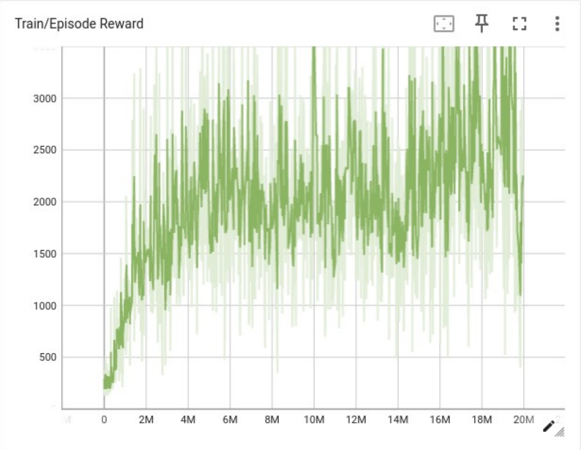
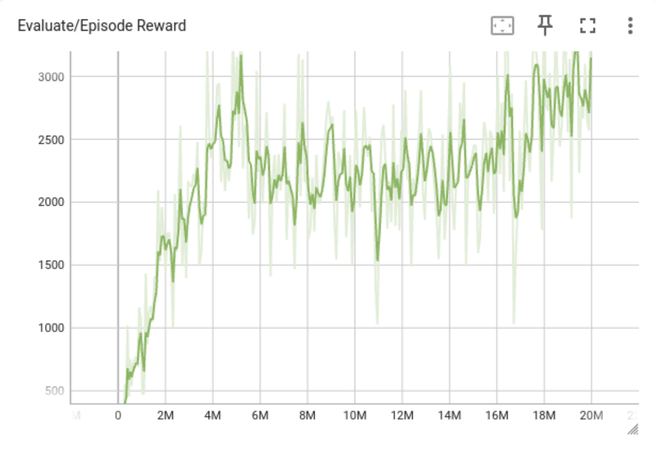
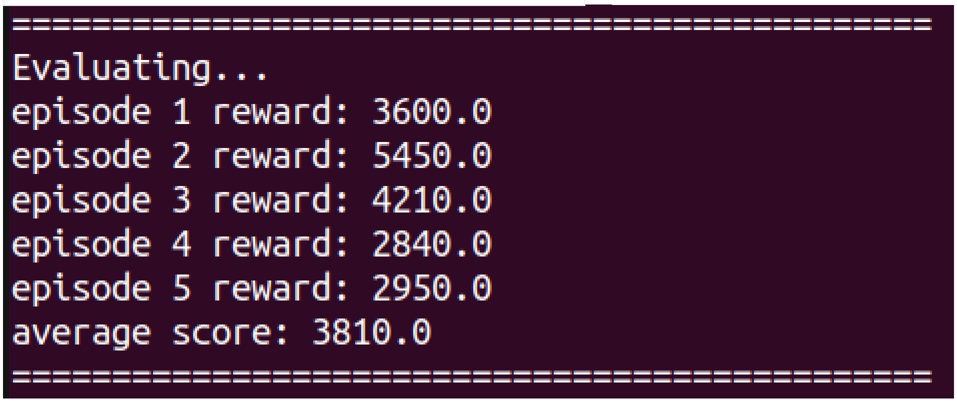
RL_LAB_2_313552041_洪日昇

Report (30% + Bonus 20%)

Screenshot of Tensorboard training curve and testing results on DQN.







```
=================================================
Evaluating...
episode 1 reward: 3600.0
episode 2 reward: 5450.0
episode 3 reward: 4210.0
episode 4 reward: 2840.0
episode 5 reward: 2950.0
average score: 3810.0
=================================================
```

Understand Deep Q-learning mechanisms.

DQN 將 Q Learning 的概念與深度學習相結合。這裡，Q 函數不像 2048 是一個簡單的表格，而是通過一個深度神經網絡來逼近。神經網絡的輸入是環境的狀態 s，輸出是所有可能行動的 Q 值。

DQN Workflow

1. Initialize the behavior network, target network, and replay buffer.
2. Interact with the environment by selecting actions using the epsilon-greedy policy.
3. Store experiences in the replay buffer.
4. Sample mini-batches from the replay buffer for training.
5. Calculate the target Q-value using the target network.
6. Compute the loss between the target Q-value and the predicted Q-value.
7. Update the behavior network using gradient descent to minimize the loss.
8. Periodically update the target network to reflect the behavior network.
9. Reduce epsilon over time to balance exploration and exploitation.
10. Repeat the process until the agent converges to an optimal policy.

Select action according to epsilon-greedy.

完全按照以前學習的步驟去跑遊戲，可能會造成有一些玩法沒有被嘗試到，除了利用學習到的知識 Exploitation，還會利用 epsilon 的機率去做 Exploration，此外因為越後面的學習是越好的，因此 Reduce epsilon over time(Epsilon Decay)。

Understand the mechanism of both behavior network and target network.
在 DQN 中，有兩個神經網絡：一個是正在訓練的 behavior network，selects actions and is trained via backpropagation. It is updated frequently.
另一個是 target network a frozen copy of the behavior network that is updated periodically,
這有助於穩定訓練過程。

Construct Q-values and target Q-values.
Q-values (Q(s, a)) are the values predicted by the behavior network for each action in the current state.
Target Q-values (Q_target) are the values the agent is trying to predict during training, computed using Bellman equation, Qtarget(s,a)=r+γa′maxQ(s′,a′).

Calculate loss function.
The loss function is usually Mean Squared Error (MSE), which measures how far the predicted Q-values are from the target Q-values:

Loss=(Q(s,a)−Qtarget(s,a))^2

The network learns by adjusting the weights to minimize this loss.

Update behavior and target network.
Perform backpropagation to compute the gradients of the loss.
Update the weights of the behavior network using the optimizer.

Understand the mechanism of experience replay buffer.
在訓練過程中，不是直接使用當前的經驗來更新網絡，而是將這些經驗存儲在，並 samples a random mini-batch of transitions from the replay buffer. This mini-batch is used to calculate the loss and update the neural network.這有助於打破資料之間的相關性，並穩定學習過程。

Learn to construct and design neural networks.
在 atari_model.py 中使用在 DRL 中常用的 CNN，self.cnn 為 Convolutional Layers 用來 Feature Extraction，self.classifier 為 Fully Connected Layers 用來 Classification/Regression。

1. Input Layer:
Input with stacked 4 grayscale frames of the Atari game screen, size (84, 84), capture temporal information, as a single frame cannot capture the dynamics of the game.
2. Convolutional Layers (Feature Extraction):
有三個 layer 用來做 Feature Extraction 每次都搭配 ReLU introduces non-linearity into the network and allows it to learn complex patterns.

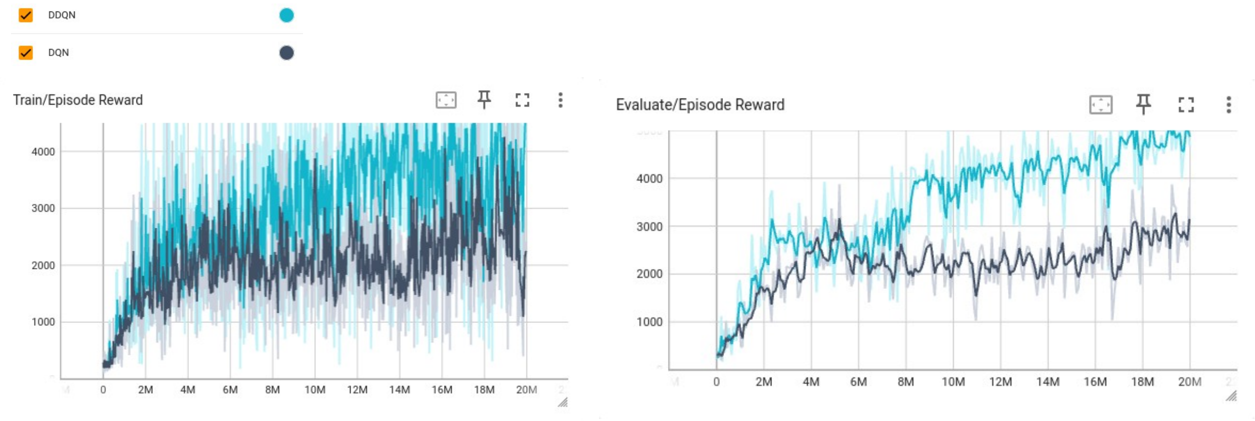
3. Flattening Layer:
After passing through the convolutional layers, the resulting feature map is flattened into a 1D vector to feed into the fully connected layers.

4. Fully Connected Layers (Decision Making):
The fully connected layers (also called classifier layers) map the extracted features to the final outputs, which are the Q-values for each action.
Layer 1: A fully connected layer with 512 units. This reduces the flattened vector from the convolutional layers to a lower-dimensional representation.
Layer 2: The final fully connected layer outputs a vector of size num_classes. Each element in this vector corresponds to the Q-value for one of the possible actions (in this case, num_classes=4 for 4 possible actions).

5. Output Layer:
The output of the network represents the Q-values for each action. In a reinforcement learning context, these values are used to select the best action using an epsilon-greedy policy or other strategies.
No activation function is applied in the output layer, as Q-values can be unbounded.

Bonus: (20%)

1.Screenshot of Tensorboard training curve and testing results on DDQN, and discuss the difference between DQN and DDQN (3%).



$$Y_t^{Q} = r_{t+1} + \gamma \max_a Q(S_{t+1}, \boxed{a} | \theta^-)$$

$$\Downarrow$$

$$Y_t^{DoubleQ} = r_{t+1} + \gamma Q\left(S_{t+1}, \boxed{\underset{a}{\operatorname{argmax}}\, Q(S_{t+1}, a|\theta)} |\theta^-\right)$$

上方的 Target Q-value Calculation 為 DQN 與 DDQN 不一樣的地方，可以發現到其實只差一點點而已，在 code 裡面也是只有差一兩行而已。

可以很明顯看到 DDQN 的表現明顯比較好。

DQN 有一個缺點就是可能會 overestimation bias，尤其在複雜的環境中會有這個問題，為了解決 overestimation，我們可以使用 DDQN 來降低。

DQN 是直接從 target network 中選取最大 Q 值，DDQN 則是從 behavior network 選取 the best action 再從 target network evaluate Q 值。