National Taiwan University
Department of Electrical Engineering
Name: Ching-Hsiang, Wu
ID: R11921080

113-1 (Fall 2024)
Reinforcement Learning

Assignment 3 Report

Q1. Try using at least two different algorithms to train your 2048 model. Present your results and discuss possible reasons why some algorithms perform better than others in this task. (12 pts.)

I tried two algorithms to train my 2048 model. They are PPO and DQN, separately. The best results of these two method are shown below,

- DQN
  - average highest : 411.6
  - average score : 4716.72
- PPO
  - average highest : 47.56
  - average score : 430.96

In the 2048 game, DQN performs better than PPO(it's worth noting that the reward setting and policy network are different between these two algorithms). I chose the DQN algorithm instead of others because the 2048 game has a discrete action space. I think it's easier to train if I choose DQN.

Q2. Show your best tile distribution in 2048. (4 pts.)

Fig. 1 shows the best tile distribution in 2048.

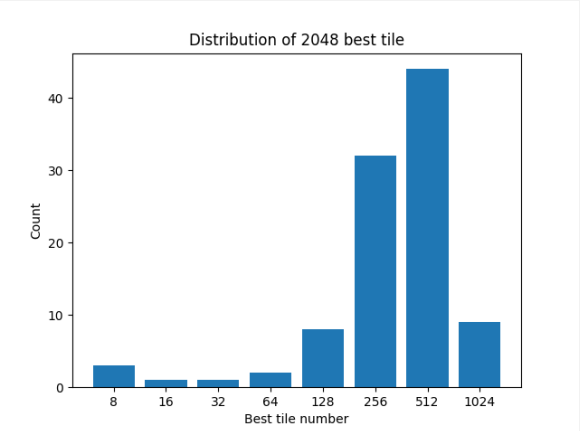


Figure 1: Best tile distribution in 2048

Q3. Describe what you have done to train your best model. (10 pts.)

My thoughts on how to train a good model to play 2048 game with **DQN** algorithm are as follows

- Tune hyperparameters
- Design a reasonable reward

- Consider a better feature extractor

To better and faster observe the training performance of each setting, I trained each setting for 50 epochs and observing the learning curve through *wandb*. Only a few settings having the best performance would be selected for more training epochs.

Firstly, without the other hyperparameter modifications, I found that increasing the learning rate helps to increase the training performance a lot. The other modification and hyperparameters' settings are listed in Table below,

Parameters	**Original**	**Modification**
learning_rate (α)	1e-4	1e-3
timesteps_per_epoch	1000	10000
exploration_fraction	0.1	0.5
target_update_interval	10000	20000

Secondly, from my personal experience of playing the 2048 game, there are some tips to achieve higher scores,

1. Stack the highest score in the corner and make the whole board monotonic, which means that we want our board to look like Fig. 2
2. The more empty tiles there are, the more chance for the player to be alive.
3. Any operation not changing the board's state is unwanted, which only wastes our time.

Figure 2: An illusion of monotonicity in 2048 game

To make the model to learn these tips, I designed my reward $\mathcal{R}$ as follows,

$$\mathcal{R} = \begin{cases} \mathcal{R}_{score} + \mathcal{R}_{weight} + \mathcal{R}_{empty} & score > 0, \\ 0 & score = 0, \\ -10 & illegalMove = True. \end{cases} \tag{1}$$

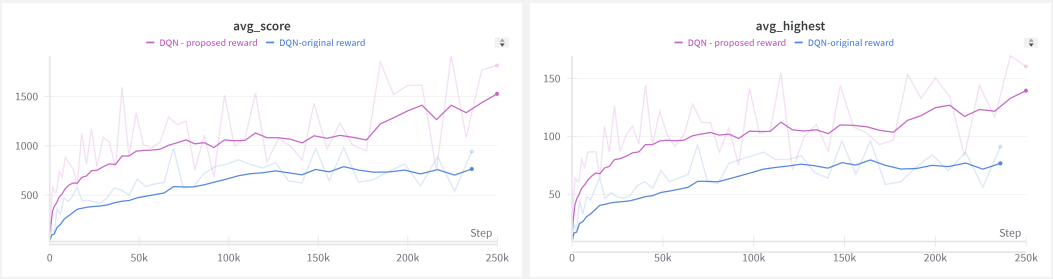
where $\mathcal{R}_{score}$ is the obtaining score when adding tile. $\mathcal{R}_{weight}$ is the extra weighting score when the board in a certain pattern, which is designed as,

$$\mathcal{R}_{weight} = \| \frac{1}{\log_2(t_b)} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ 4 & 3 & 2 & 1 \end{bmatrix} .* M_b \| \tag{2}$$

where t_b is the highest number of tiles in the current board and M_b is the current board state matrix. $\mathcal{R}_{empty}$ is the extra reward for empty tiles, which is designed as,

$$\mathcal{R}_{empty} = \frac{\log_2(t_b)}{10} \cdot E_n \tag{3}$$

where E_n is the number of empty tiles. The maximum number of illegal moves in my environment is 10. The comparison between the default reward and the proposed reward is shown in Fig. 3



(a) Average score comparison (b) Average highest comparison

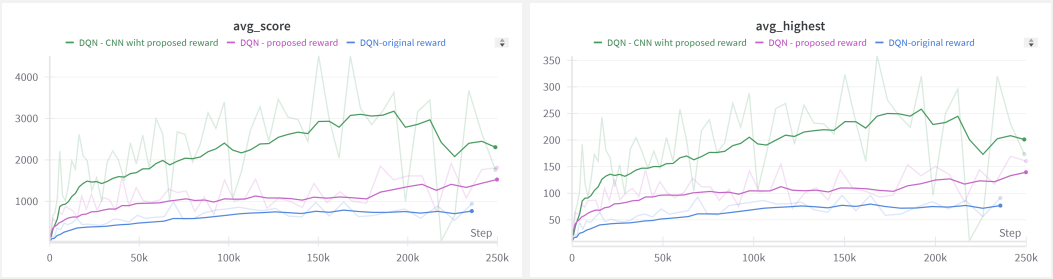
Figure 3: The comparison of original and proposed rewards in average and highest scores.

Thirdly, even if I designed a better reward function, I still cannot achieve more than 512 of the highest tiles. As a result, inspired by the discussion on *Slack*, I decided to design my own CNN feature extractor. The developed procedure is referred to *Stable-Baseline 3* policy network tutorial. The shape of the designed feature extractor is shown in Fig. 4

```
(features_extractor): CNN2048(
  (cnn): Sequential(
    (0): Conv2d(16, 64, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU()
    (2): Flatten(start_dim=1, end_dim=-1)
  )
  (linear): Sequential(
    (0): Linear(in_features=256, out_features=64, bias=True)
    (1): ReLU()
  )
)
```

Figure 4: Customized CNN feature extractor architecture.

A single convolution layer along with 64 kernels was selected. The kernel size is (3×3) because I want to extract a certain pattern with a large range. The results after applying the customized CNN feature extractor, named as **CNN2048** in the following report, are shown in Fig. 5



(a) Average score comparison (b) Average highest comparison

Figure 5: The comparison of original and proposed rewards under CNN2048.

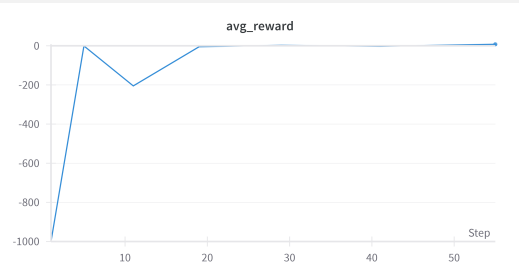
The performance after applying CNN2048 is way better than the original linear feature extractor.

Lastly, I trained the DQN with CNN2048 on my 2048 game environment with a designed reward function with 500 epochs to obtain the best model, which can achieve 1024 9 times out of 100 rollouts!

Q4. Choose an environment from the Gymnasium library and train an agent. Show your results or share anything you like. (10 pts.)

The environment I choose is *Lunar Lander*. The action is to control three engines, right, left, and main, to make the lunar lander land safely on the landing pad.

I train it with the default PPO algorithm. After training 5 epochs, the reward curve looks like it is converging, however, the lander didn't land on the landing pad, and it just hovered in the air instead. The reward curve and training result are shown in Fig. 6



(a) Accumulated reward for 10 rollouts:5 epochs

(b) The training result: lunar lander hovering

Figure 6: The training result for 5 epochs.

It's an interesting result and here I try to give appropriate explanations. From Fig. 6a, we know that the reward is always negative and converges to zero. The problem is in the default reward function, the agent will receive -100 if the lander doesn't land safely. I think it will make the agent choose to hover in the air rather than risk its safety to get -100 punishments.

If we can add more punishing rewards on hovering or add more extra rewards once the lander touches the ground will make it more willing to close to the landing pad.

After training for over 15 epochs, the agent find out that landing on the pad can obtain 100 reward and it finally did it. Fig. 7 shows the final training result.

(a) Accumulated reward for 10 rollouts:15 epochs

(b) The training result: lunar lander landing

Figure 7: The training result for 15 epochs.