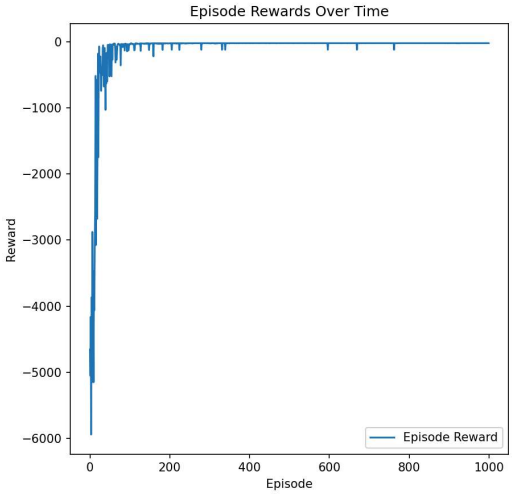
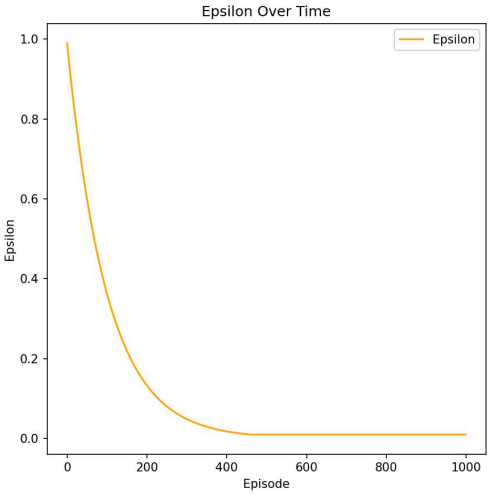
Report of Homework 3

1. Result Visualization of Cliff-walk Game

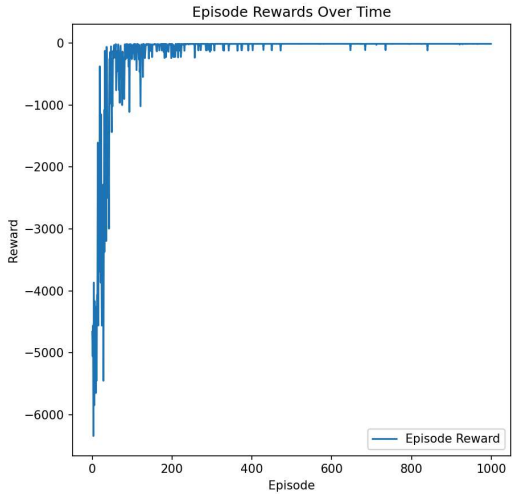
1.1 episode reward and ϵ-value during the training process

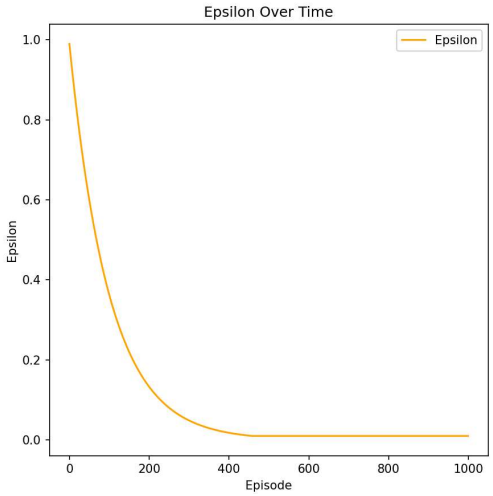
SARSA



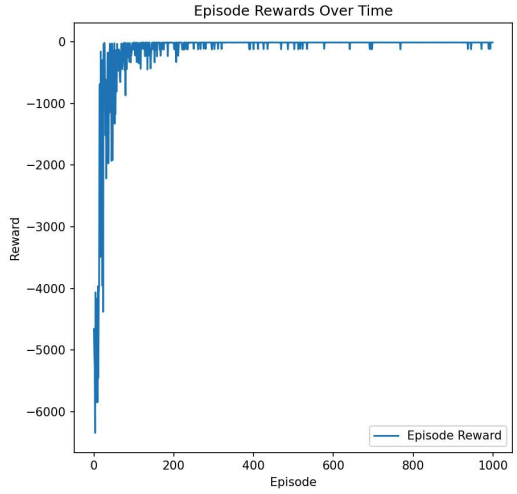


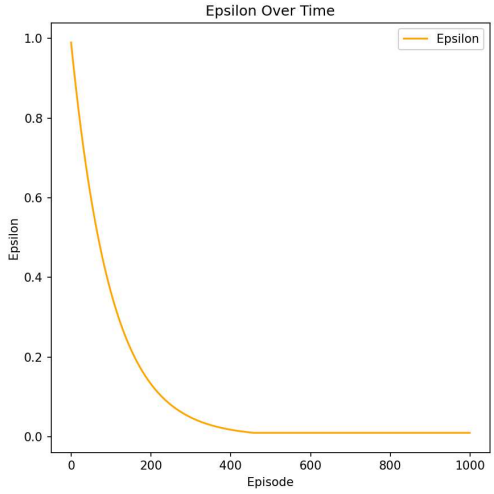
Q-Learning





Dynamic Q-Learning

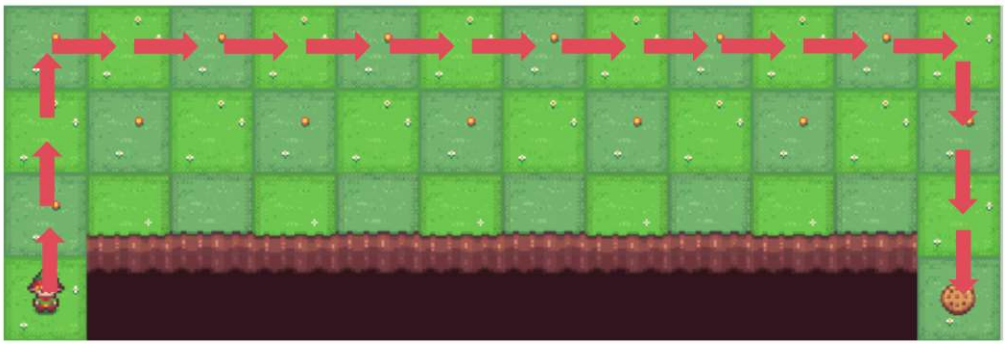




1.2 Visualize the final paths found by the intelligent agents after training

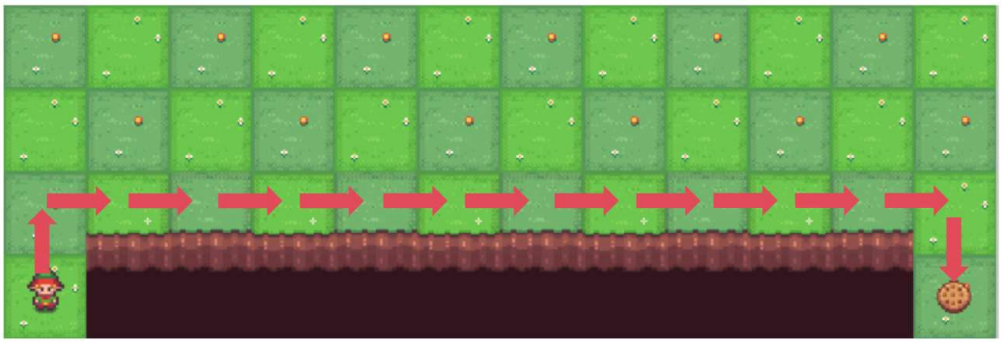
The final paths found by the agents after training can be found in document "videos".

SARSA



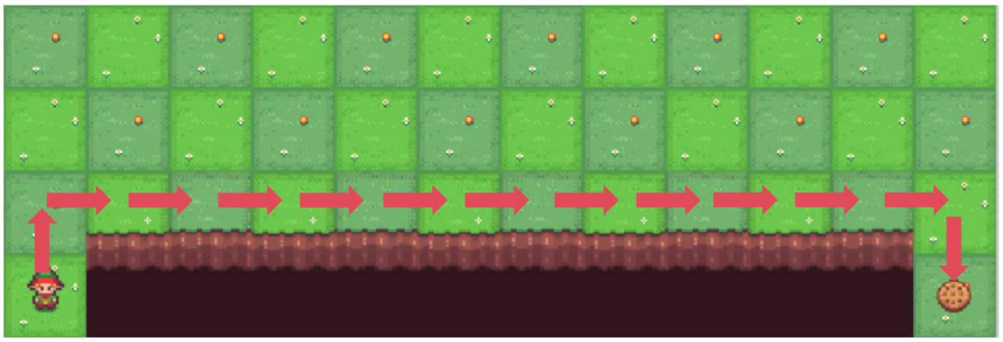
```
The video of testing results with agent trained by SARSA can be found in videos/sarsa.
```

Q-Learning



```
The video of testing results with agent trained by SARSA can be found in videos/q-learning.
```

Dynamic Q-Learning



```
The video of testing results with agent trained by SARSA can be found in videos/dyna-q.
```

2. Result Analysis of Cliff-walk Game

2.1 the difference between the paths found by Sarsa and Q-learning and the reason

2.1.1 Different paths

The agent trained by Sarsa **goes up three steps**, then turn right, and after reaching the edge, goes down three steps to reach the terminal. The agent trained by Q-learning goes up one step, then turn right(**along the edge of cliff**), and after reaching the edge, goes down one step to reach the terminal.

2.1.2 The underlying logic of two algorithms

SARSA is a method based on an "on-policy" strategy that considers actions and their outcomes in the current state to update the value function. At each step, SARSA uses the current strategy to determine what should be done next. Therefore, in the problem of cliff walking, if SARSA's initial strategy is to fall into the cliff, it may tend to choose a path away from the cliff, as this reduces the risk of immediate fall. Since SARSA updates the values on a complete trajectory under the current policy, it is more likely to converge to a safe and stable path, such as going to the second line.

Q-Learning is an "off-policy" method because it does not rely on any specific strategy to update the value function. On the contrary, Q-Learning always chooses actions with the highest expected return, even if it means taking risks close to a cliff. This is because Q-Learning considers the maximum expected return on all possible follow-up actions, rather than following a specific strategy. In this case, Q-Learning may find that the path closer to the

cliff provides higher immediate rewards (such as reaching the target faster), although this involves greater risk. Therefore, the strategy learned by Q-Learning may be to walk along the edge of a cliff, as this path may provide the maximum cumulative reward in the long run.

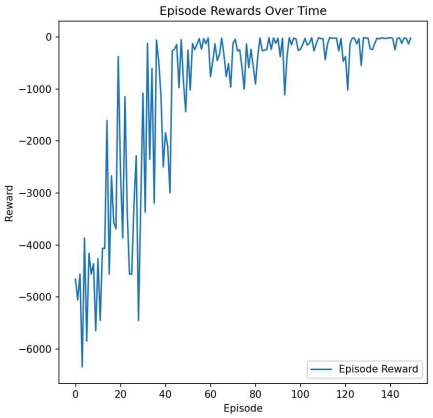
2.1.3 Summarize the reason

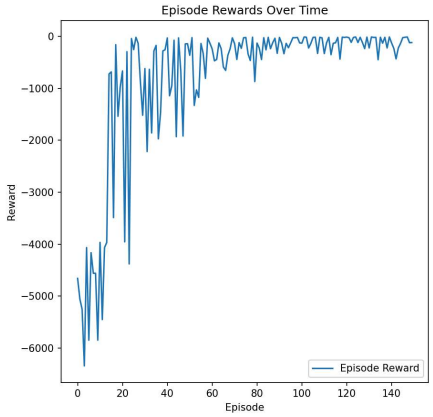
In summary, SARSA places greater emphasis on the safety and stability of current strategies, while Q-Learning seeks to maximize potential long-term benefits, even if it involves assuming certain risks. That's why after training process, SARSA learns the path to be far away from the cliff, while Q-Learning chooses the path to walk along the cliff.

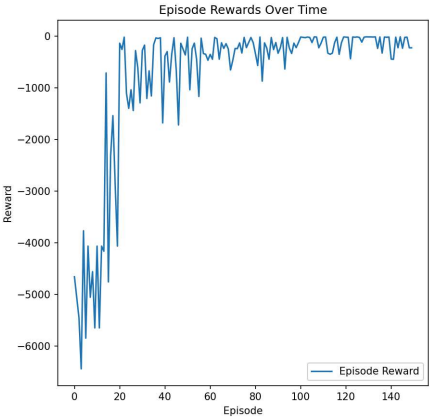
2.2 Analysis of the training efficiency between model-based RL and model-free alorithms

2.1.1 Training Efficiency

The following images show the variation of rewards with the number of training rounds(within 150) for Q-planning steps of 0, 10, and 20, respectively.







As the number of Q-planning steps increases(The Q-planning steps of Q-Learning is 0), the convergence speed of the Dyna-Q algorithm increases. That is to say, the training efficiency getss faster as the number of Q-planning increases.

2.1.2 Reason Analysis

初始化$Q(s,a)$，初始化模型$M(s,a)$

for 序列$e=1 \to E$ **do**:

  得到初始状态s

  for $t=1 \to T$ **do**:

    用ϵ-贪婪策略根据Q选择当前状态s下的动作a

    得到环境反馈的r, s'

    $Q(s,a) \leftarrow Q(s,a)+\alpha[r+\gamma \max_{a'} Q(s',a')-Q(s,a)]$

    $M(s,a) \leftarrow r, s'$

    for 次数 $n=1 \to N$ **do**:

      随机选择一个曾经访问过的状态s_m

      采取一个曾经在状态s_m下执行过的动作a_m

      $r_m, s'_m \leftarrow M(s_m, a_m)$

      $Q(s_m,a_m) \leftarrow Q(s_m,a_m)+\alpha[r_m+\gamma \max_{a'} Q(s'_m,a')-Q(s_m,a_m)]$

    end for

    $s \leftarrow s'$

  end for

end for

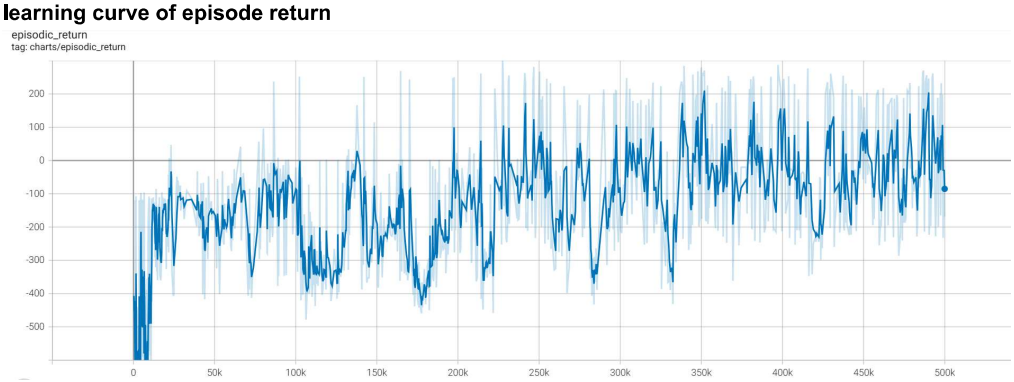
From the baseline of Dyna-Q algorithm shown above, we can see that it has N steps of interaction with the enviroment based on Q-Learning algorithm, which contributes to higher accuracy of Q-table. Dyna-Q uses Q-planning to generate simulated data based on the model, and then improves the strategy using both simulated and real data.

3. Train and Tune the DQN Agent

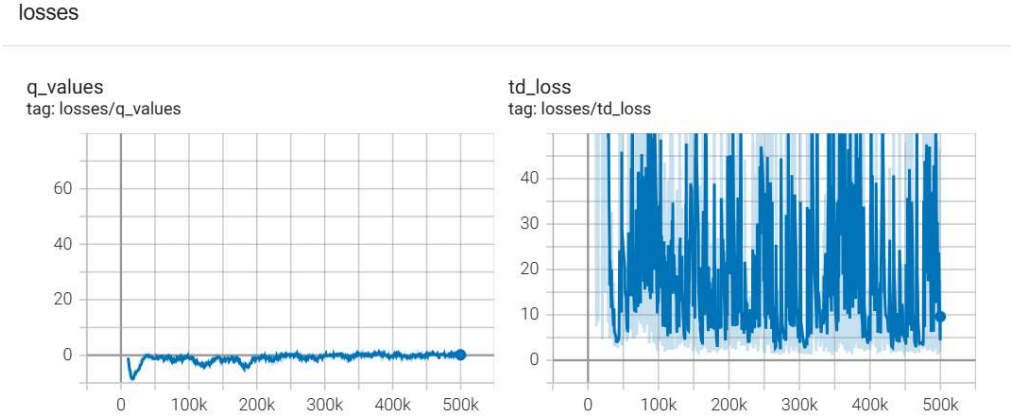
3.1 The performance of agent trained with the given parameters

3.1.1 Training process

learning curve of episode return



learning curve of losses



3.1.2 Testing results

testing results of 50 rounds

```
Episode 1: Total Reward = -20.160007355665556
Episode 2: Total Reward = -20.073826177071766
Episode 3: Total Reward = -44.92164508187871
Episode 4: Total Reward = -247.56291376672087
Episode 5: Total Reward = -219.73340304331126
Episode 6: Total Reward = -235.64781031147513
Episode 7: Total Reward = -113.99430518057164
Episode 8: Total Reward = 232.7152119063174
Episode 9: Total Reward = -280.2641070936636
Episode 10: Total Reward = -213.99760698084225
Episode 11: Total Reward = -219.6538778627564
Episode 12: Total Reward = -250.28289230360807
Episode 13: Total Reward = -196.88562550901784
Episode 14: Total Reward = -266.1319641043889
Episode 15: Total Reward = 205.59231840821158
Episode 16: Total Reward = 237.12913522902625
Episode 17: Total Reward = -22.626605439905234
Episode 18: Total Reward = -39.67988760690494
Episode 19: Total Reward = 168.1971371214853
Episode 20: Total Reward = 177.74984365800552
Episode 21: Total Reward = -195.69573663748218
Episode 22: Total Reward = -34.086901400307056
Episode 23: Total Reward = -218.71296848575648
Episode 24: Total Reward = -222.74926568657528
Episode 25: Total Reward = 212.55447964890186
Episode 26: Total Reward = -69.40774954122985
Episode 27: Total Reward = -5.877721723457981
Episode 28: Total Reward = 221.53272205861623
Episode 29: Total Reward = -18.924020413862678
Episode 30: Total Reward = -163.824104650845
Episode 31: Total Reward = -61.80548464750658
Episode 32: Total Reward = 258.83559353763997
Episode 33: Total Reward = -33.887477229055904
Episode 34: Total Reward = 189.23776794659767
Episode 35: Total Reward = -213.2446079295961
Episode 36: Total Reward = -195.98699362009216
Episode 37: Total Reward = -186.41227961740088
Episode 38: Total Reward = 213.45576408119172
Episode 39: Total Reward = -194.1959269207415
Episode 40: Total Reward = -98.76127656853743
Episode 41: Total Reward = -213.8489042231068
Episode 42: Total Reward = -183.96823186865234
Episode 43: Total Reward = -273.44421367781536
Episode 44: Total Reward = 257.85781906681325
Episode 45: Total Reward = -196.008396584121
Episode 46: Total Reward = 11.37921764604313
Episode 47: Total Reward = 226.6774660099578
Episode 48: Total Reward = -191.13991891626574
Episode 49: Total Reward = -211.81809096764283
Episode 50: Total Reward = 194.99691222876302
Mean Reward = -55.35010721160526
```

The video of testing results with old parameters can be found in videos/dqn_test_old_params.

3.2 The performance of agent trained with the our parameters

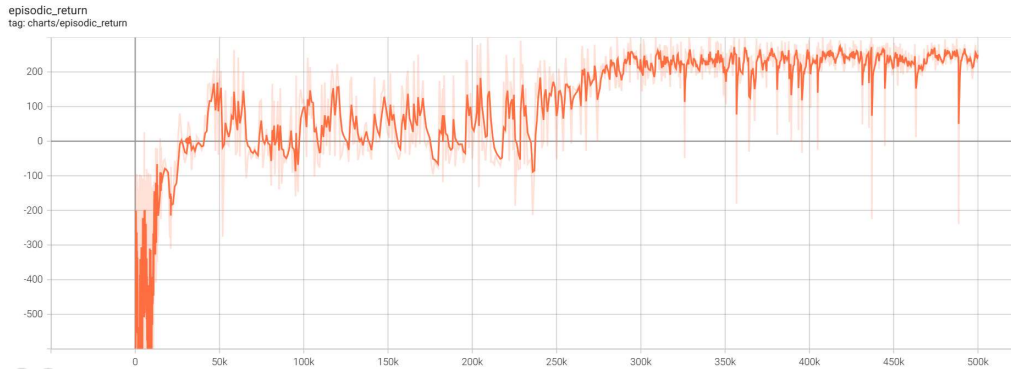
3.2.1 Our parameters of DQN model

Parameter	given Value	Our Value	how to design
total-timesteps	500000	500000	
learning-rate	2.5e-4	1e-3	Add learning rate to accelerate learning process
buffer-size	10000	50000	increase the buffer size to learn from more experience
gamma	0.6	0.99	increase discount factor to decrease time influence

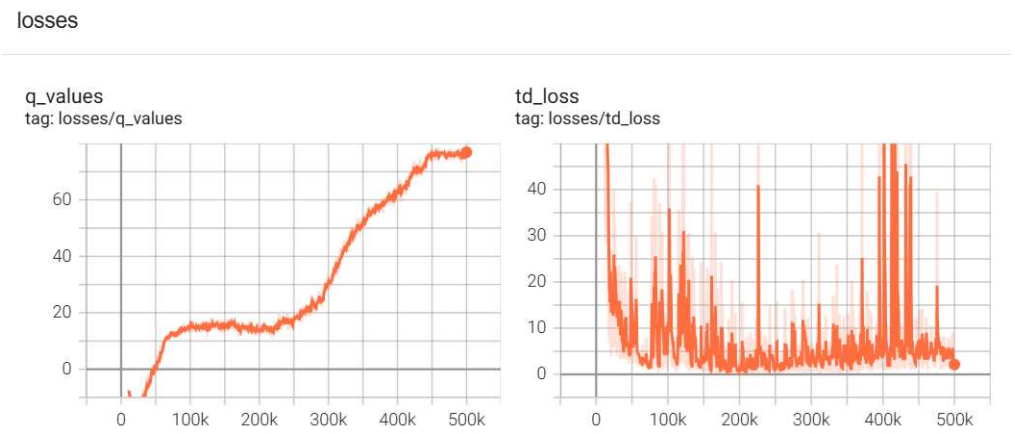
Parameter	given Value	Our Value	how to design
target-network-frequency	500	250	Reduce the frequency of target network updates to accelerate learning progress
batch-size	128	256	increase batch size to make learning more comprehensive
start-e	0.3	0.2	
end-e	0.05	0.02	
exploration-fraction	0.1	0.3	Increase the duration of the exploration phase
learning-starts	10000	10000	
train-frequency	10	8	

3.2.2 Training process

learning curve of episode return



learning curve of losses



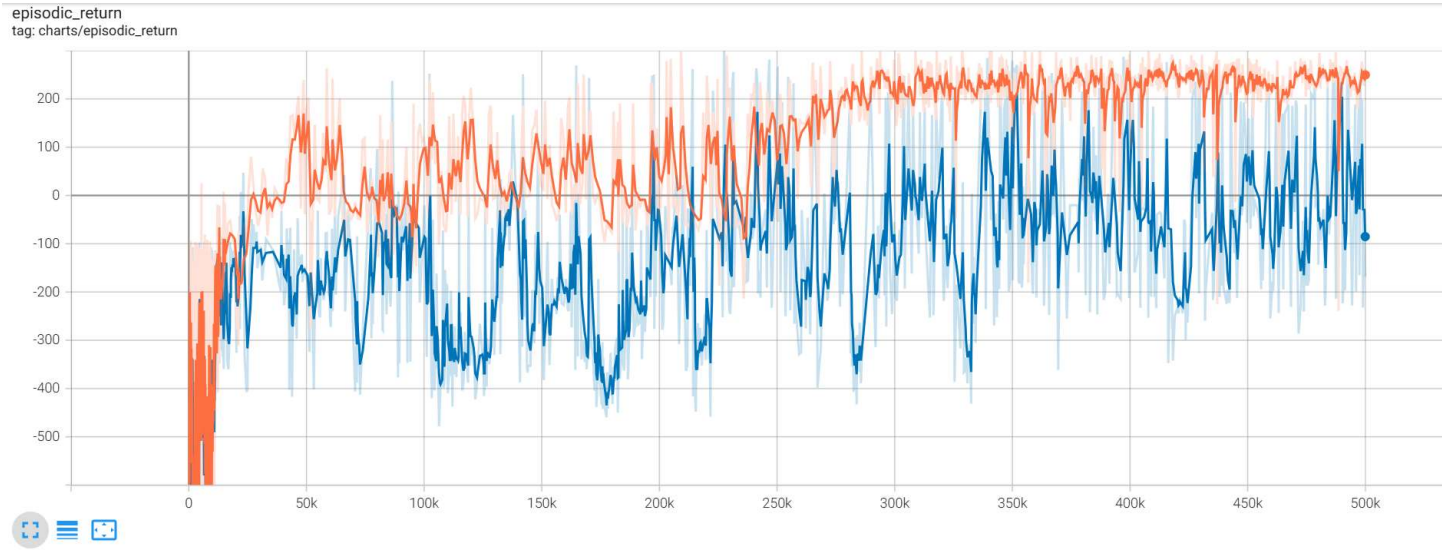
3.2.3 Testing results

testing results of 50 rounds

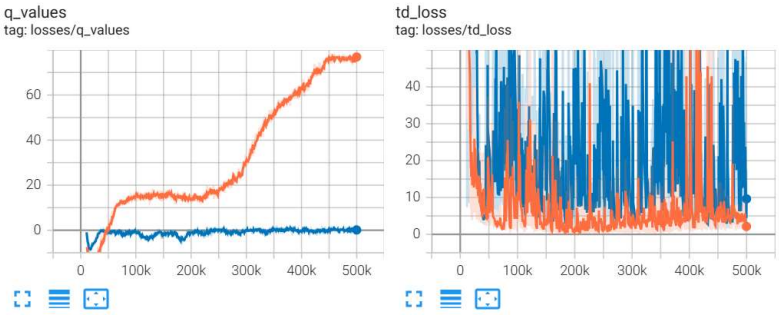
```
Episode 1: Total Reward = 290.85452975923135
Episode 2: Total Reward = 228.99977223334668
Episode 3: Total Reward = 248.72611696509443
Episode 4: Total Reward = 227.7912420583121
Episode 5: Total Reward = 275.20039731121835
Episode 6: Total Reward = 288.88609977910085
Episode 7: Total Reward = 279.6703163093986
Episode 8: Total Reward = 286.12962467714294
Episode 9: Total Reward = 247.5065026854005
Episode 10: Total Reward = 237.29913559977618
Episode 11: Total Reward = 282.76322925639175
Episode 12: Total Reward = 262.60781822446864
Episode 13: Total Reward = 268.37444791335395
Episode 14: Total Reward = 257.93805850526076
Episode 15: Total Reward = 207.0952848147458
Episode 16: Total Reward = 243.70968900322708
Episode 17: Total Reward = 253.68913337575546
Episode 18: Total Reward = 228.00416714753126
Episode 19: Total Reward = 229.502072031621
Episode 20: Total Reward = 224.4052599126669
Episode 21: Total Reward = 266.5520306160199
Episode 22: Total Reward = 201.57923865570268
Episode 23: Total Reward = 274.0395385212656
Episode 24: Total Reward = 250.12392994632307
Episode 25: Total Reward = 290.8839116705411
Episode 26: Total Reward = 268.59582001656474
Episode 27: Total Reward = 260.90975815764307
Episode 28: Total Reward = 234.13463345182055
Episode 29: Total Reward = 239.5675689366659
Episode 30: Total Reward = 289.9640432522681
Episode 31: Total Reward = 225.23426917568764
Episode 32: Total Reward = 242.7619153083321
Episode 33: Total Reward = 257.5387382532993
Episode 34: Total Reward = 249.2124213830399
Episode 35: Total Reward = 260.73933319711466
Episode 36: Total Reward = 230.81284722742825
Episode 37: Total Reward = 264.8695690422377
Episode 38: Total Reward = 271.8849548439143
Episode 39: Total Reward = 223.0341423926369
Episode 40: Total Reward = 277.6999626815391
Episode 41: Total Reward = 291.7404409716838
Episode 42: Total Reward = 259.274948481268
Episode 43: Total Reward = 212.77540873899736
Episode 44: Total Reward = 239.5192430020382
Episode 45: Total Reward = 260.09334716713715
Episode 46: Total Reward = 264.40158220714414
Episode 47: Total Reward = 261.1994875453476
Episode 48: Total Reward = 222.3389951179318
Episode 49: Total Reward = 272.94633313976794
Episode 50: Total Reward = 232.50732410784352
Mean Reward = 253.321772695445
```

The video of testing results with our parameters can be found in videos/dqn_test_new_params.

3.2.4 Comparison of Training Processes for Models Corresponding to New and Old Parameters



losses



4. Improve Exploration Schema: UCB(Upper Confidence Bound) Algorithm

4.1 Introduction of UCB Algorithm

UCB(Upper Confidence Bound) Algorithm is a classic uncertainty based strategy algorithm.

It is based on a famous inequality:

Hoeffding's inequality

Suppose $X_1, X_2, \dots, X_n \in [0,1]$ are n independent and identically distributed random variables.
The experience expectation is $\bar{x}_n = \frac{1}{n}\sum_{k=1}^{n} X_k$,
then we have $P\{E[X] \geq \bar{x}_n + u\} \leq e^{-2nu^2}$.

Let us apply Hoeffding's inequality to exploration problem.
Let $\bar{x}_n$ be $\hat{Q}_t(a)$(The expectation reward of action a found by us).
Let $u = \hat{U}_t(a)$(The metric of uncertainty of action a).
By Hoeffding's inequality, given a probability $p = e^{-2N_t(a)U_t(a)^2}$($N_t(a)$ is the number of times action a is taken), $Q_t(a) < \hat{Q}_t(a) + \hat{U}_t(a)$ holds at least with probability $1-p$.($Q_t(a)$ is the actual expectation reward of action a)

When p is sufficiently small, $Q_t(a) < \hat{Q}_t(a) + \hat{U}_t(a)$ holds a high probability of being true.
$\hat{Q}_t(a) + \hat{U}_t(a)$ can be seen as the upper bound of reward expectation.
UCB algorithm take the action with the largest upper bound of reward expectation, that is $a = argmax_{a \in A}[\hat{Q}_t(a) + \hat{U}_t(a)]$.($\hat{U}_t(a) = (\frac{-logp}{2N_t(a)})^{\frac{1}{2}}$)

Therefore, after setting a probability p, we can calculate the metric of uncertainty $\hat{U}_t(a)$.
Intuitively, the UCB algorithm first estimates the upper bound of the expected reward for each action before selecting it, so that there is only a small probability that the expected reward for each action will exceed this upper bound. Then, it selects the action with the highest upper bound of the expected reward, and thus chooses the action that is most likely to obtain the maximum expected reward.

4.2 Evaluation of UCB Algorithm

4.2.1 Merits

1. The UCB algorithm effectively balances the relationship between exploration and utilization by calculating the upper bound of the confidence interval for each option. This enables the algorithm to explore the unknown without ignoring the optimal options already known.

2. The algorithm can automatically adapt to changes in the environment without the need for manual intervention to adjust parameters.
3. The UCB algorithm is relatively simple, easy to understand and implement, and does not require complex computing resources.

4.2.2 Drawbacks

1. Although the UCB algorithm can quickly discover the optimal action, in order to pursue the confidence level of each action, the convergence speed to the optimal action is slower than some methods, such as softmax.
2. The UCB algorithm is more difficult to extend to more general reinforcement learning environments than the ϵ-greedy algorithm, and it is difficult to solve non-stationary problems (where the benefits of actions do not come from a stationary probability distribution) and problems with massive state spaces.