BATCH-CONSTRAINED DEEP Q-LEARNING(BCQ) ON D4RL AND ABLATION STUDY

111550049 林德倫

111550151 徐嘉亨 111550177 吳定霖

INTRODUCTION

Extrapolation error :

- Offline RL settings
- Mismatch between batch and current policy
- Q value unaccurate value estimation

How BCQ solve it :

- Fixed the current policy without mismatching the state-action pairs in the batch.

INTRODUCTION

Techniques in BCQ :

- Use VAE to generate action most likely in the batch.
- Use a perterbation model to increase diversity of actions.
- Use Clipped Double Q-learning to penalize uncertainty.

Algorithm 1 BCQ

Input: Batch $\mathcal{B}$, horizon T, target network update rate τ, mini-batch size N, max perturbation Φ, number of sampled actions n, minimum weighting λ.
Initialize Q-networks $Q_{\theta_1}, Q_{\theta_2}$, perturbation network ξ_ϕ, and VAE $G_\omega = \{E_{\omega_1}, D_{\omega_2}\}$, with random parameters θ_1, θ_2, ϕ, ω, and target networks $Q_{\theta'_1}, Q_{\theta'_2}$, $\xi_{\phi'}$ with $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \phi' \leftarrow \phi$.
for $t = 1$ **to** T **do**
  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\mu, \sigma = E_{\omega_1}(s, a), \quad \tilde{a} = D_{\omega_2}(s, z), \quad z \sim \mathcal{N}(\mu, \sigma)$
  $\omega \leftarrow \text{argmin}_\omega \sum (a - \tilde{a})^2 + D_{\text{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$
  Sample n actions: $\{a_i \sim G_\omega(s')\}_{i=1}^n$
  Perturb each action: $\{a_i = a_i + \xi_\phi(s', a_i, \Phi)\}_{i=1}^n$
  Set value target y (Eqn. 13)
  $\theta \leftarrow \text{argmin}_\theta \sum (y - Q_\theta(s, a))^2$
  $\phi \leftarrow \text{argmax}_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$
  Update target networks: $\theta'_i \leftarrow \tau\theta + (1 - \tau)\theta'_i$
  $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
end for

MAIN APPROACH

We do six modifications on original BCQ:

1. Replace VAE with conditional GAN(CGAN)
2. Convert Clipped Double Q-learning to Clipped Quadruple Q-learning
3. Make Actor and Critic share the first layer
4. Remove the Actor(perturbation model)
5. Change discount factor to 0.9(origin is 0.99)
6. Change batch size to 200(origin is 100)

MAIN APPROACH - 1
CGAN

Method: Replace the variational auto-encode (VAE) with conditional generative adversarial net (CGAN)

Target: Try to use CGAN to get better performance

MAIN APPROACH - 2
CLIPPED QUADRUPLE Q-LEARNING

Method: Replace the modified Clipped Double Q-learning with modified Quadruple Q-learning (i.e. we will need 4 Q-networks instead of 2)

Target: We hope it make the total reward more consistent than original BCQ.

$$y = r + \gamma \max_{a_i} \left[\lambda \min_{j=1,2} Q_{\theta'_j}(s', a_i) + (1-\lambda) \max_{j=1,2} Q_{\theta'_j}(s', a_i) \right]$$

theirs

$$y = r + \gamma \max_{a_i} \left[\lambda \min_{j=1,2,3,4} Q_{\theta'_j}(s', a_i) + (1-\lambda) \max_{j=1,2,3,4} Q_{\theta'_j}(s', a_i) \right]$$

ours

MAIN APPROACH - 3
SHARED FIRST LAYER

Method: Make the Actor and Critic shared the first layer of neural network

Target: In the network, we can spend less computation time and parameters

but still reach the same performance compared with the original BCQ

MAIN APPROACH - 4
REMOVE PERTURBATION

Method: Let BCQ directly use the action generated by VAE.

Target: By removing the perturbation in the actor, we can reduce computation costs and make the model to choose the action most likely in the batch VAE determined.

MAIN APPROACH - 5
CHANGE DISCOUNT

Method: Change discount factor from 0.99 to 0.9

Target: We try a different gamma value to find the potential better model.

MAIN APPROACH - 6
CHANGE BATCHSIZE

Method: Change the batch size from 100 to 200

Target: We hope the model perform better by increasing batch size

EVALUATION

```
# Runs policy for X episodes and returns average reward
# A fixed seed is used for the eval environment
def eval_policy(policy, env_name, seed, eval_episodes=10):
        eval_env = gym.make(env_name)
        eval_env.seed(seed + 100)

        avg_reward = 0.
        for _ in range(eval_episodes):
                state, done = eval_env.reset(), False
                while not done:
                        action = policy.select_action(np.array(state))
                        state, reward, done, _ = eval_env.step(action)
                        avg_reward += reward

        avg_reward /= eval_episodes

        print("---------------------------------------")
        print(f"Evaluation over {eval_episodes} episodes: {avg_reward:.3f}")
        print("---------------------------------------")
        return avg_reward
```

Evaluation Function: same as the evaluation function from the BCQ code on GitHub

Final Results: average evaluation over 3 seeds on each dataset

EVALUATION METRIC OF STEP2

	BCQ(origin)	BCQ(CGAN)	BCQ(quadruple)	BCQ(shared)	BCQ(no perturbation)	BCQ(gamma 0.9)	BCQ(batch size 200)
hopper-random	326.5	304.4	325.0	309.55	303.3	315.0	326.6
hopper-medium	1713.2	251.7	1266.2	1151.016	987.2	899.7	1815.5
hopper-expert	3470.3	202.4	3498.3	1917.694	3500.1	2449.8	3166.6
hopper-medium-replay	925.6	134.3	962.1	987.233	918.0	302.9	908.9
hopper-medium-expert	3590.3	299.3	3489.7	3332.071	3555.4	245.4	3614.7
walker2d-random	243.8	313.2	233.7	222.877	105.7	250.5	239.0
walker2d-medium	2468.6	1187.0	2304.5	2367.923	962.5	941.5	2545.9
walker2d-expert	3826.4	325.6	4246.1	3553.091	3376.2	3804.5	3948.1
walker2d-medium-replay	871.7	719.7	695.7	755.087	483.2	455.1	631.8
walker2d-medium-expert	2275.3	491.1	2299.0	2633.07	1984.3	1183.0	2321.1
maze2d-umaze	82.3	36.2	79.4	2.4	66.8	45.4	85.0
maze2d-medium	46.5	21.3	92.8	1.5	86.3	31.2	52.0
maze2d-large	74.1	175.0	157.8	5.9	97.2	8.4	116.2
antmaze-medium-diverse	0.0	0.0	0.0	0.0	0.0	0.0	0.0
antmaze-medium-play	0.1	0.1	0.0	0.0	0.0	0.0	0.1

EVALUATION METRIC OF STEP3

	BCQ(origin)	BCQ(CGAN)	BCQ(quadruple)	BCQ(shared)	BCQ(no perturbation)	BCQ(gamma 0.9)	BCQ(batch size 200)
pen-human	1811.9	-0.2	1695.1	1716.8	1587.4	1802.4	2163.7
pen-cloned	573.1	215.4	1318.5	754.4	492.4	926.1	549.1
pen-expert	3651.3	1029.8	3621.7	3532.9	3669.7	3407.0	4010.2
hammer-human	-139.9	-248.2	-218.3	-114.8	53.5	-171.6	-226.1
hammer-cloned	-228.2	-232.7	-227.8	-185.7	-148.9	30.9	-227.7
hammer-expert	10659.3	-194.8	15280.5	14408.3	15728.6	13432.0	15096.3
door-human	-58.6	-58.6	-60.5	-55.8	25.5	-49.5	-59.5
door-cloned	-56.3	-62.0	-56.1	-55.9	-52.6	-54.4	-57.8
door-expert	2945.4	-46.5	2767.5	2820.2	3023.5	2904.9	2918.3
relocate-human	-8.4	-14.9	-7.9	-8.0	-5.9	-7.6	-9.3
relocate-cloned	-17.3	-17.8	-17.8	-17.6	-16.9	-16.5	-17.5
relocate-expert	1998.5	-18.6	1830.1	2081.8	4391.7	2107.8	2601.3

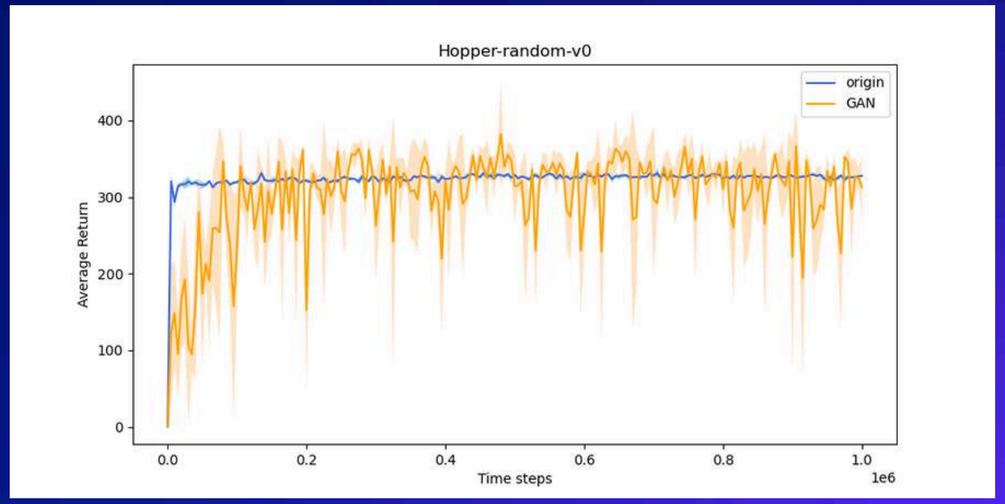
LIMITATION & PROBLEM

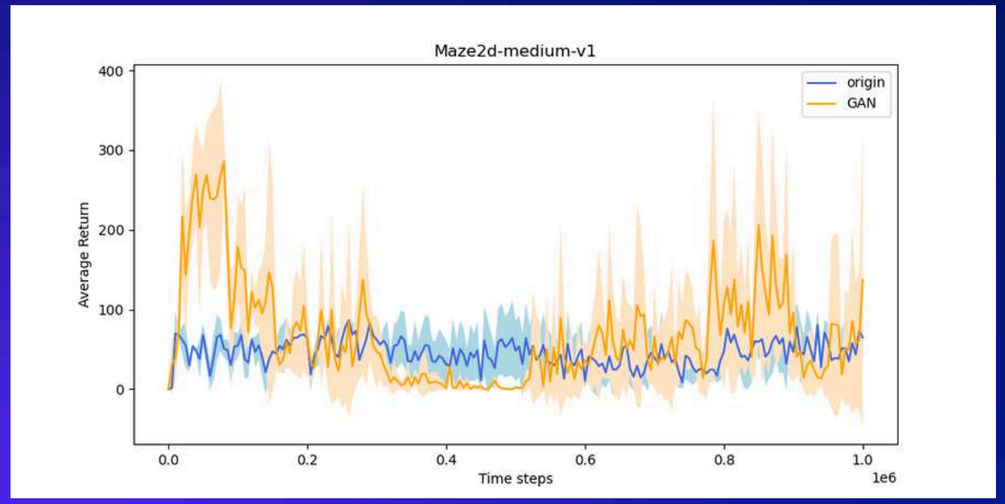
Although the use of VAE and Perturbation model in BCQ addresses the problem of extrapolation error in offline settings, it still struggles to accurately fit some state-action pairs that are out-of-distribution (OOD) from the behavior policy distribution.

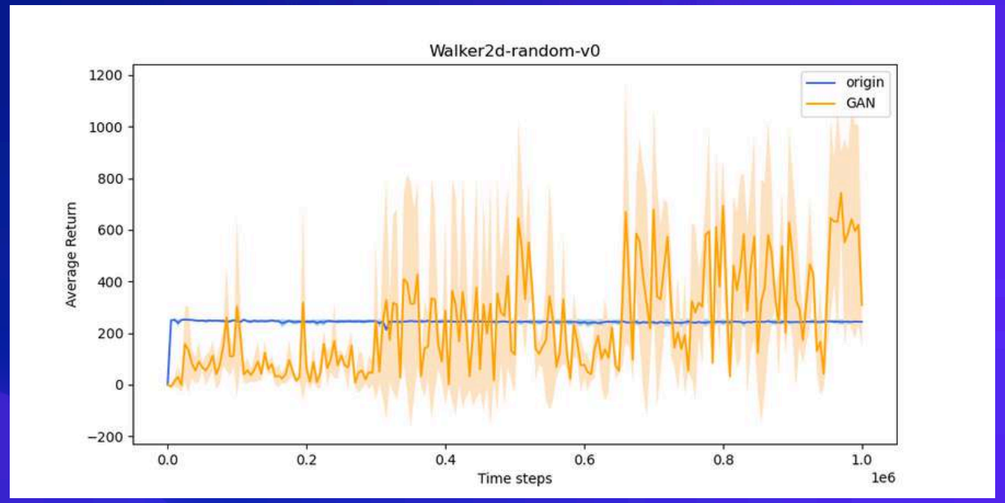
RESULTS OF STEP2

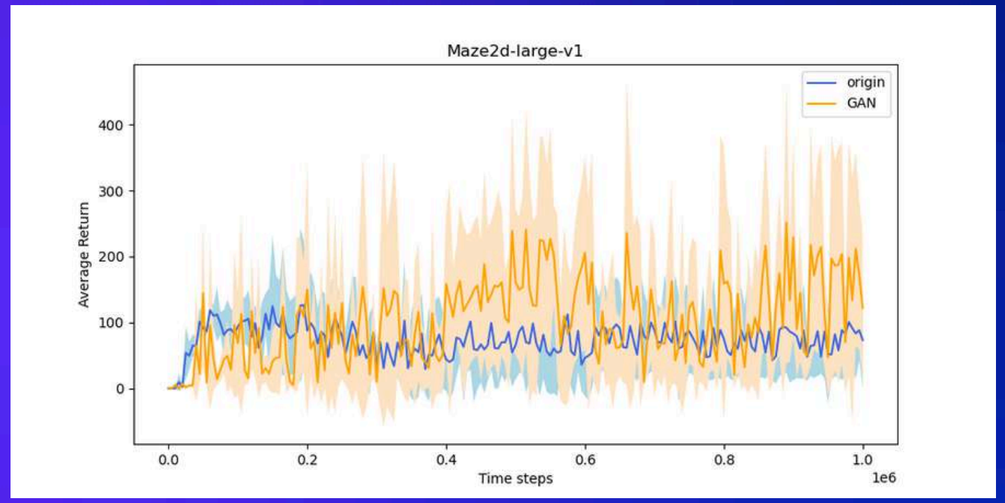
CGAN

CGAN has more exploration than the original BCQ.





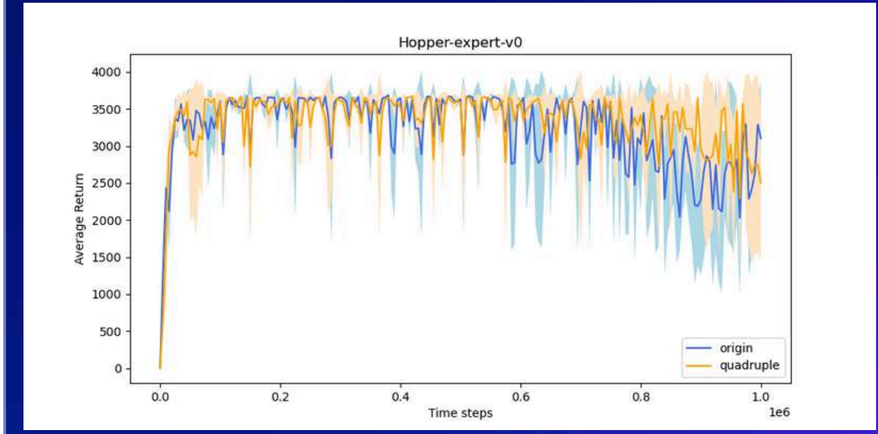


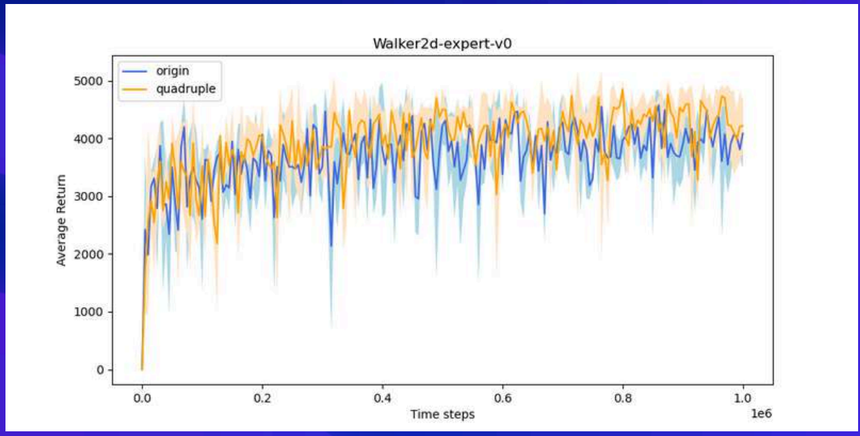


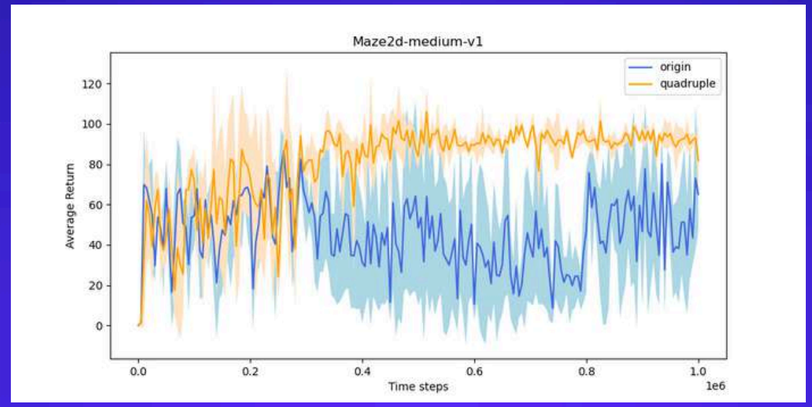
RESULTS OF STEP2

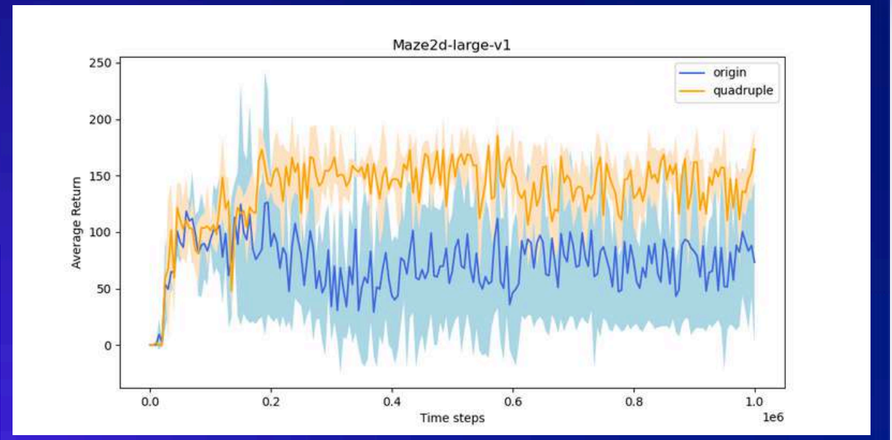
QUADRUPLE

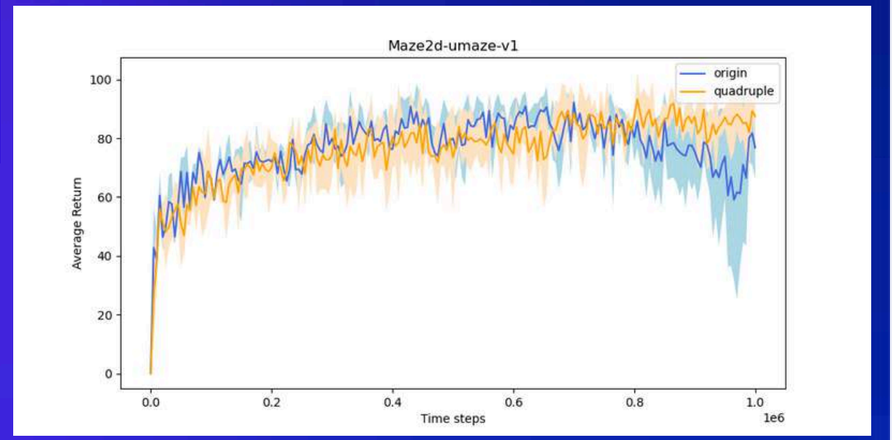
The use of Clipped Quadruple Q-learning makes the total reward more consistent than original BCQ.





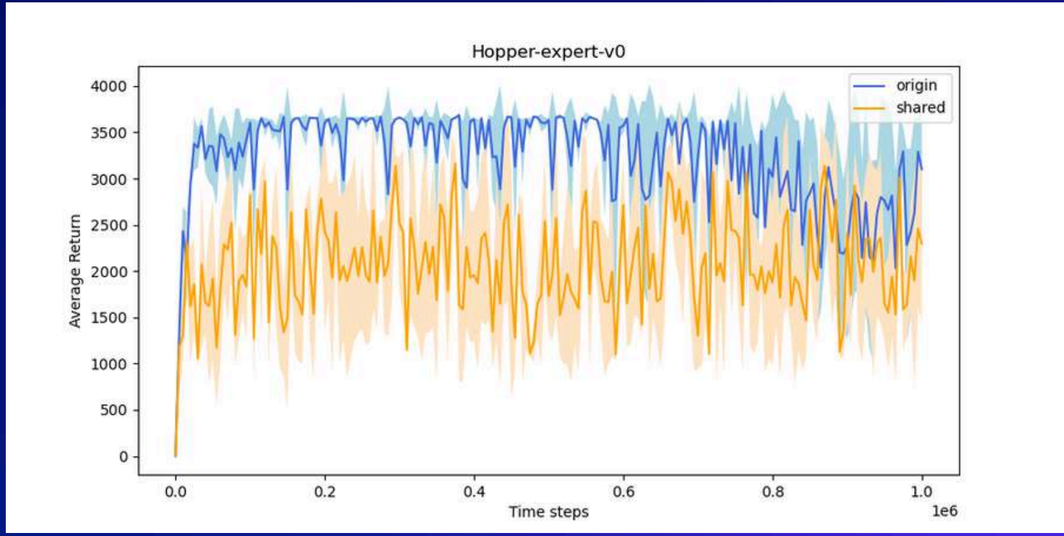


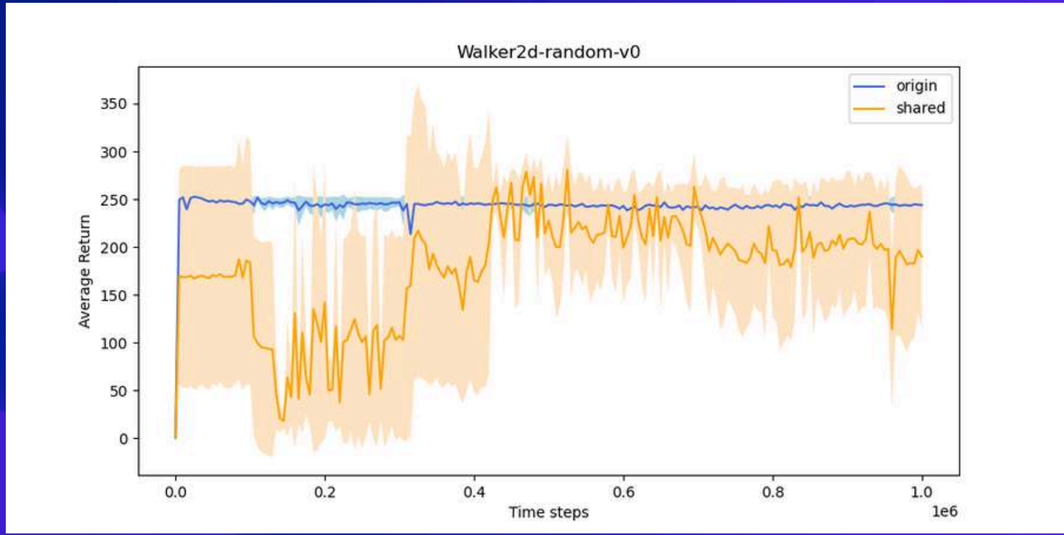




RESULTS OF STEP2

SHARED





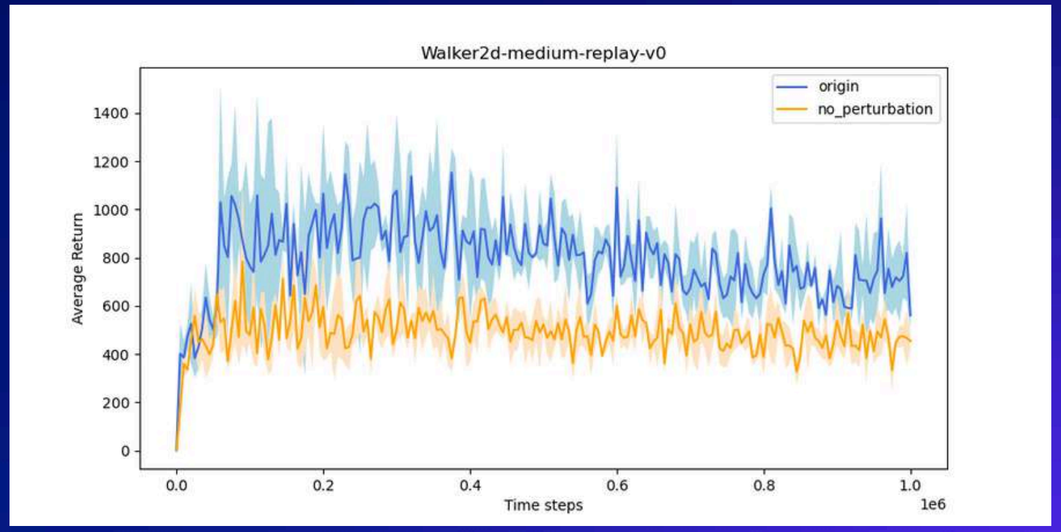
Dataset that original BCQ outperform the shared method:

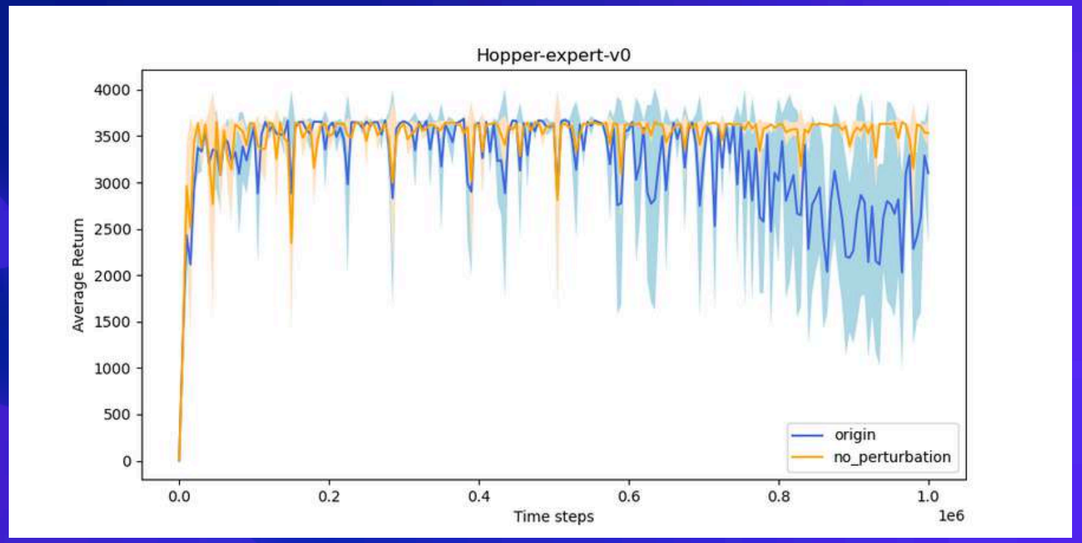
hopper-expert and walker2d-random

Reason: usage of shared layer and reducing parameters has probability to reduce the performance.

RESULTS OF STEP2

REMOVE PERTERBATION

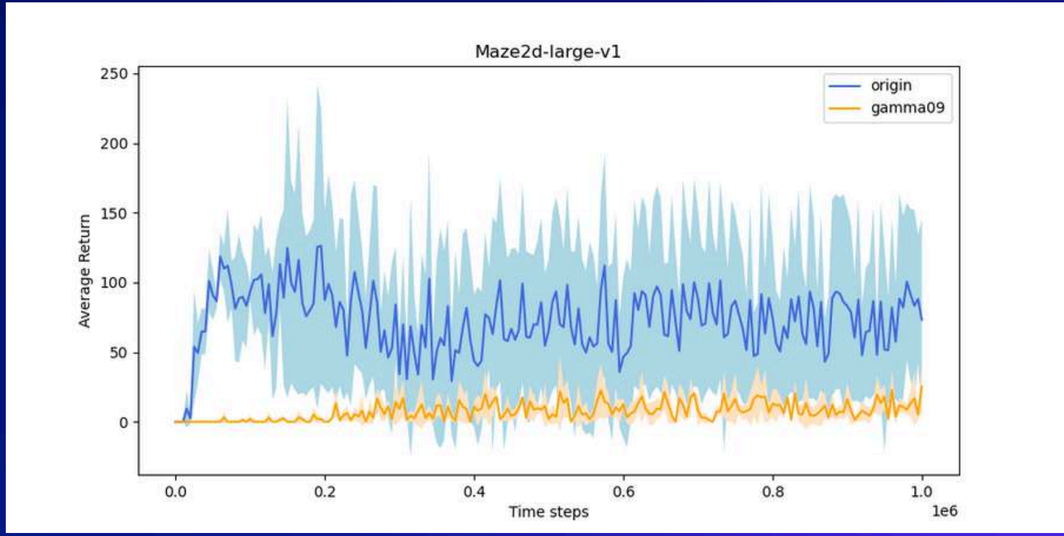


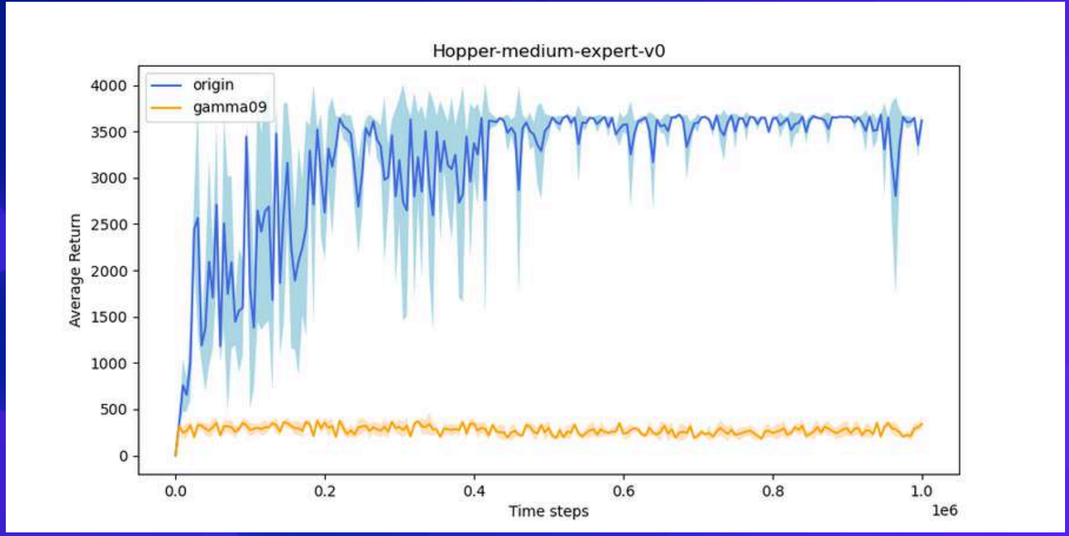


In the environment that the distribution are more random, the performances are weaker because it lacks of explorations.

However, in a expert environment, the performances are better because it is more restrict on expert batch.

RESULTS OF STEP2
CHANGE DISCOUNT

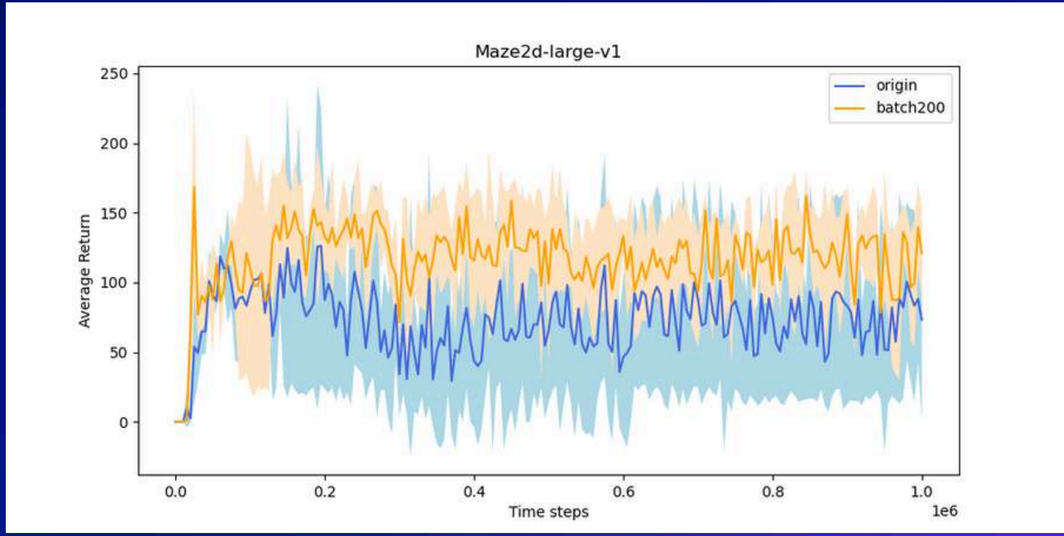




Almost every model get weaker.

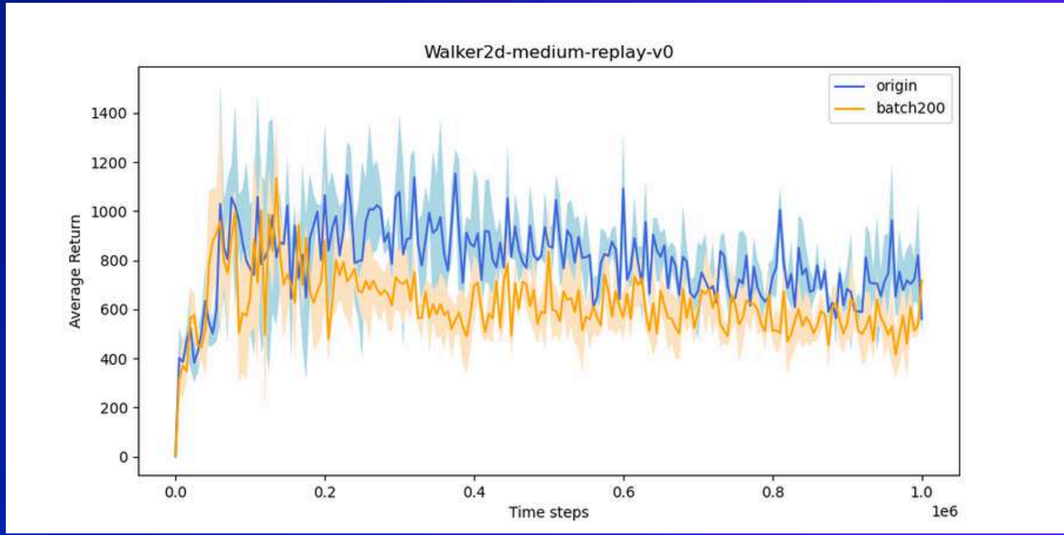
It seems that 0.99 is the proper discount value.

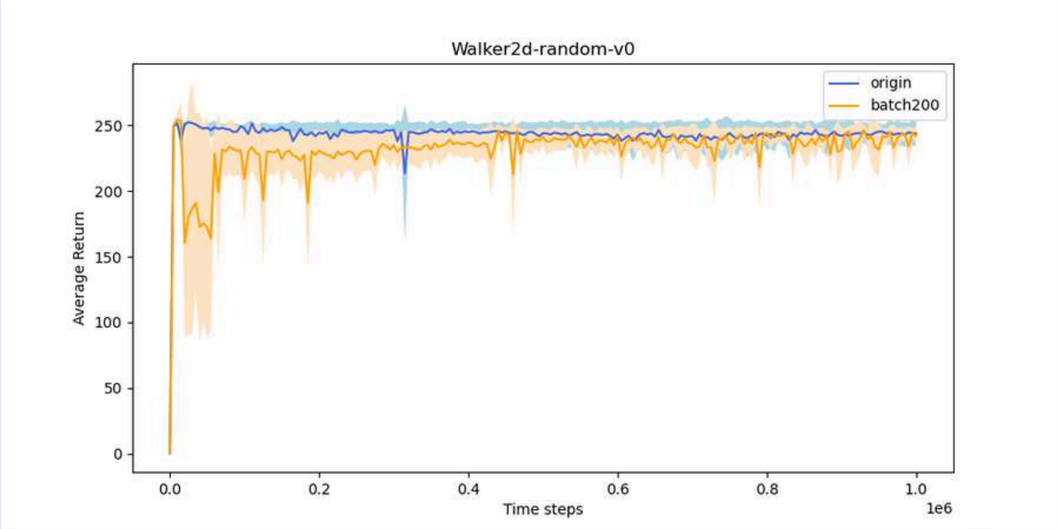
RESULTS OF STEP2
INCREASE BATCH SIZE



This change perform better then original on the task such as maze.

However, it get worse on the other task, especially those which rewards were limited.

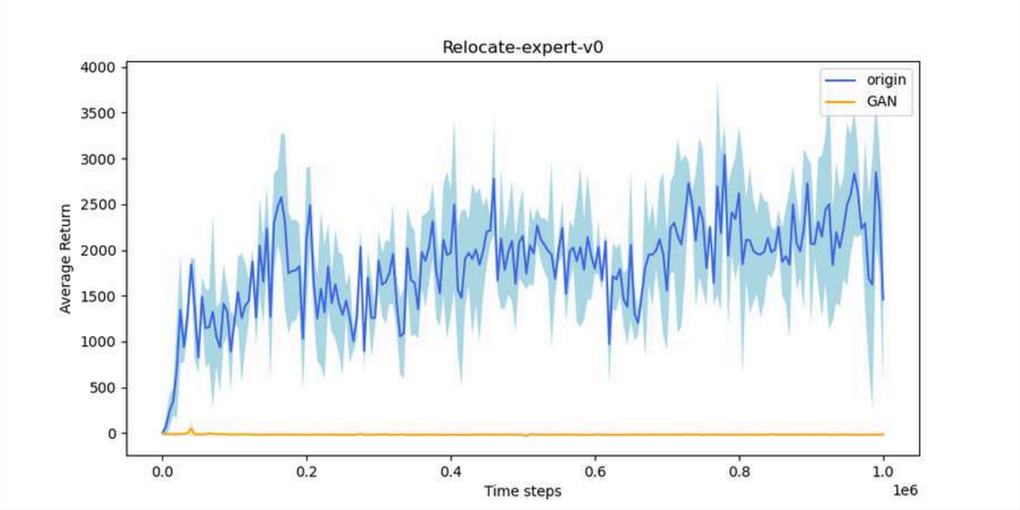


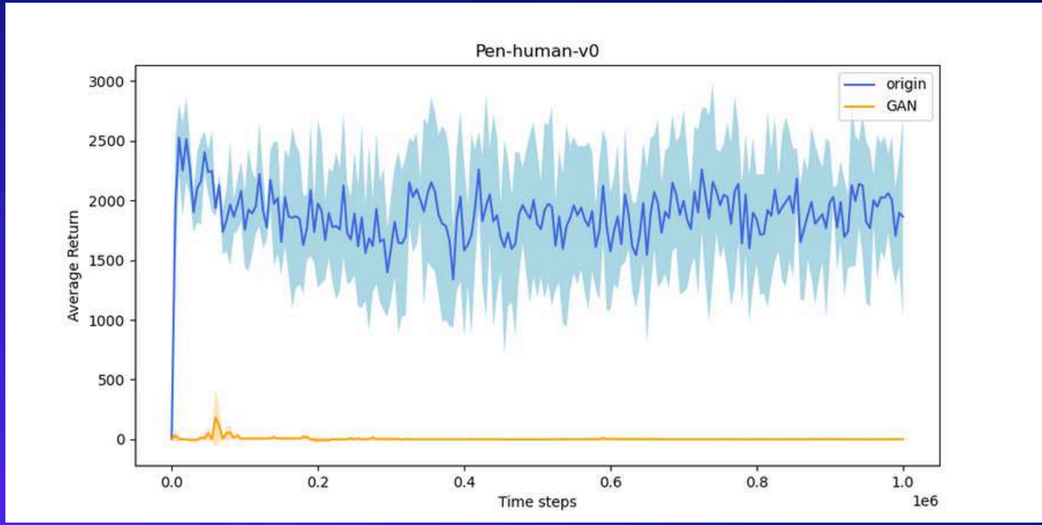


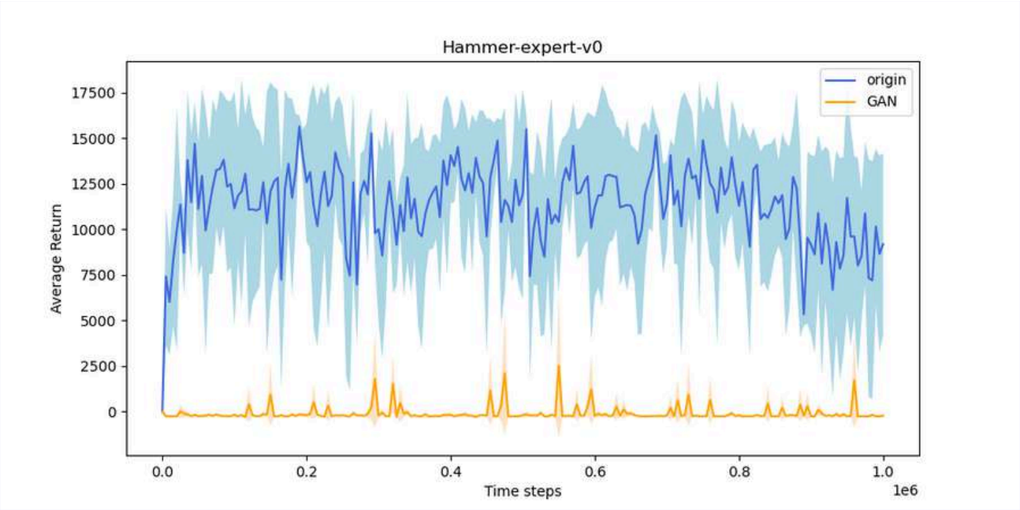
RESULTS OF STEP3

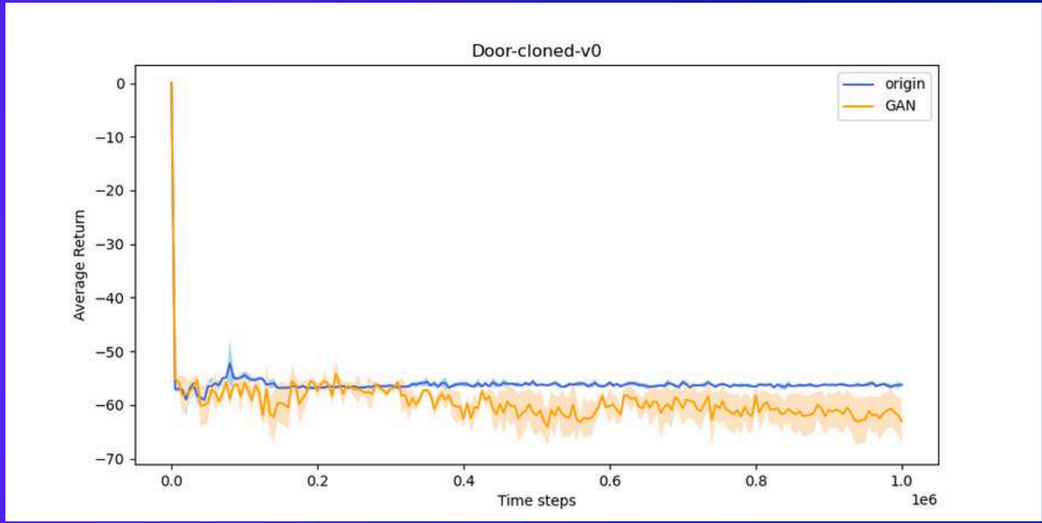
CGAN

Exploration doesn’t help solve more complex tasks(Adroit)





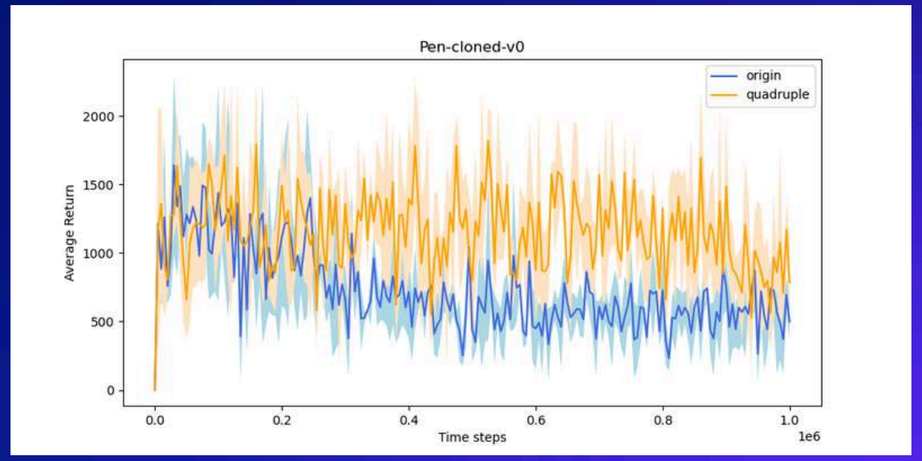


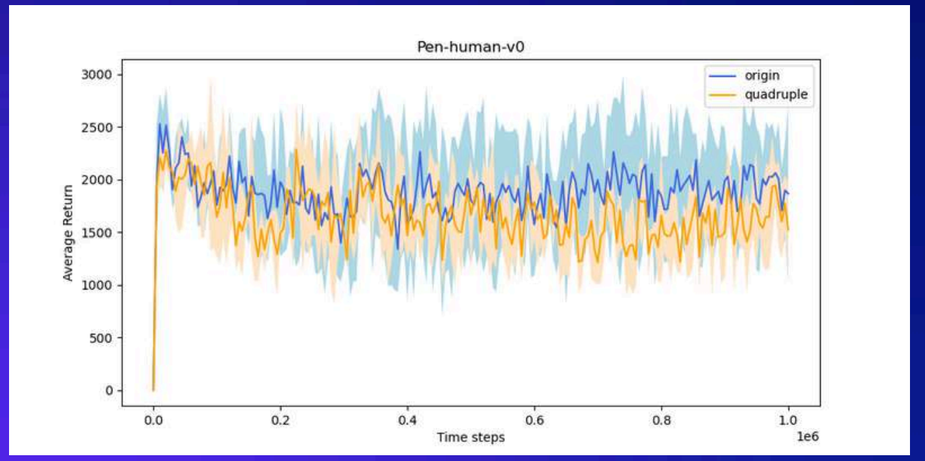


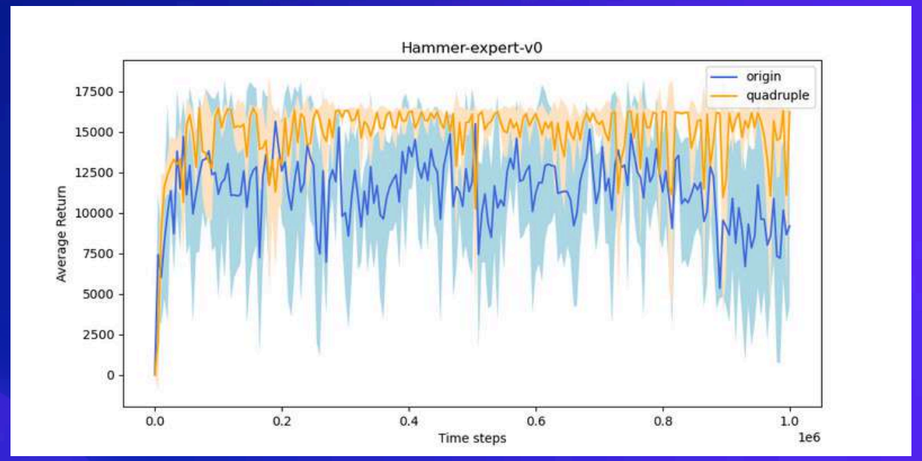
RESULTS OF STEP3

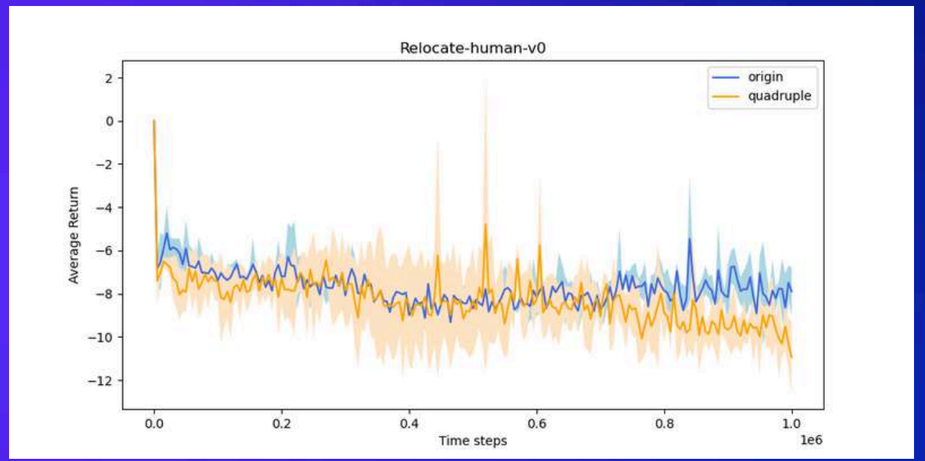
QUADRUPLE

The use of Clipped Quadruple Q-learning makes the total reward more consistent than original BCQ.





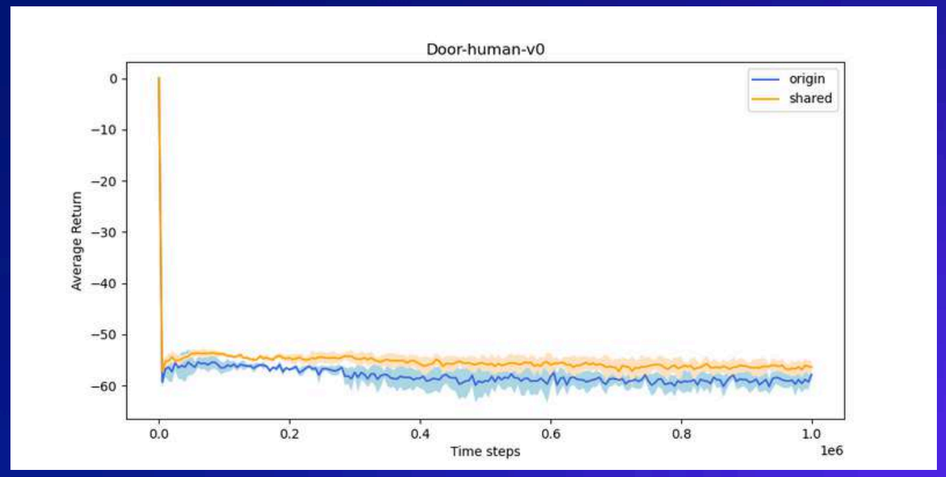


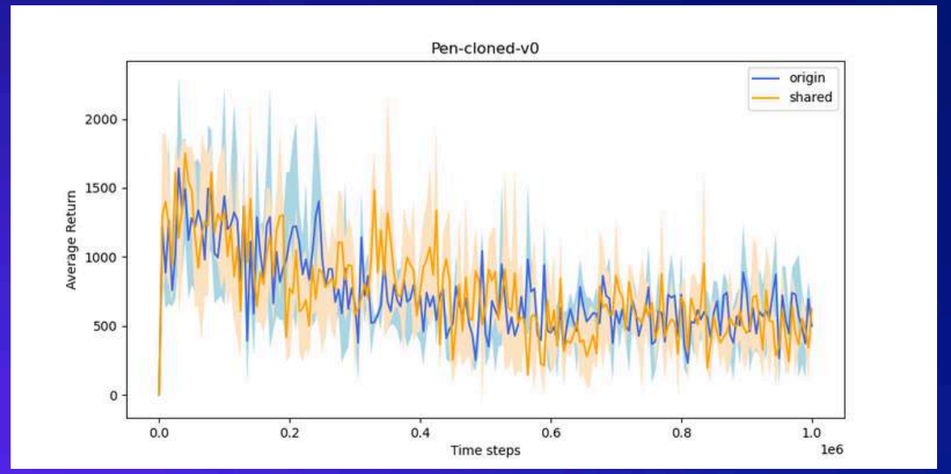


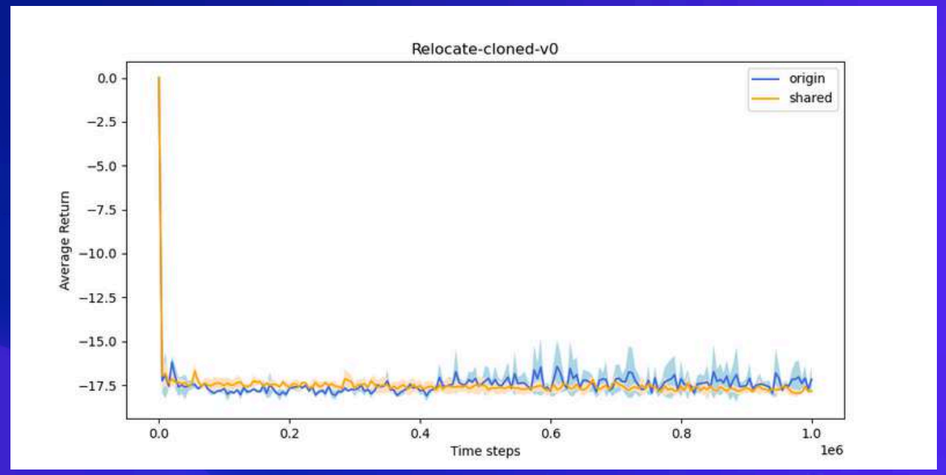
RESULTS OF STEP3

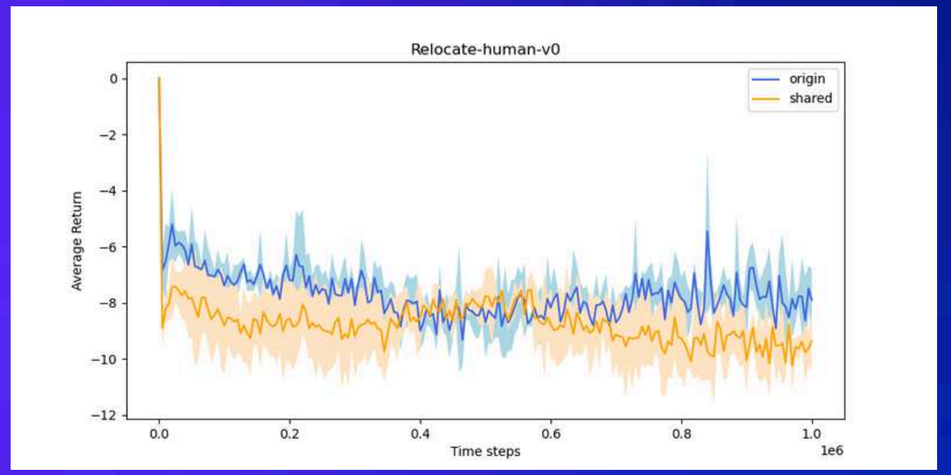
SHARED

The use of shared layer reduce the parameter, so in more complex tasks, it can be trained faster. But it performs weak after enough training epchos.





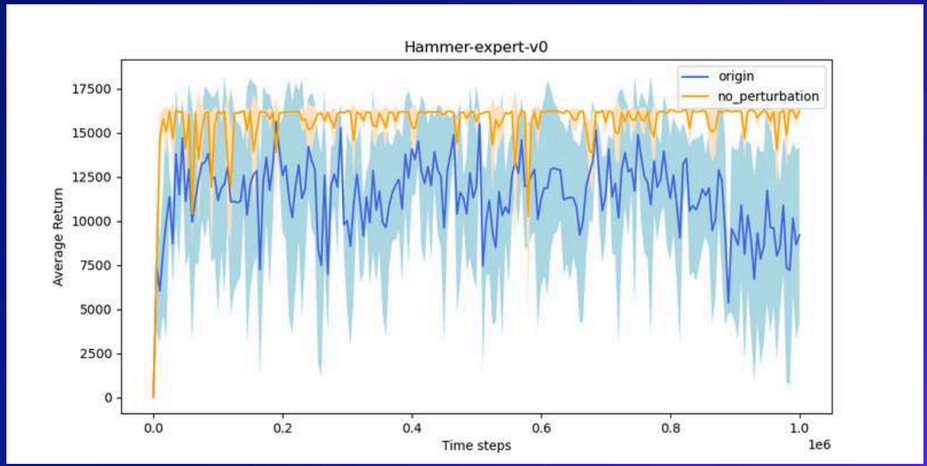


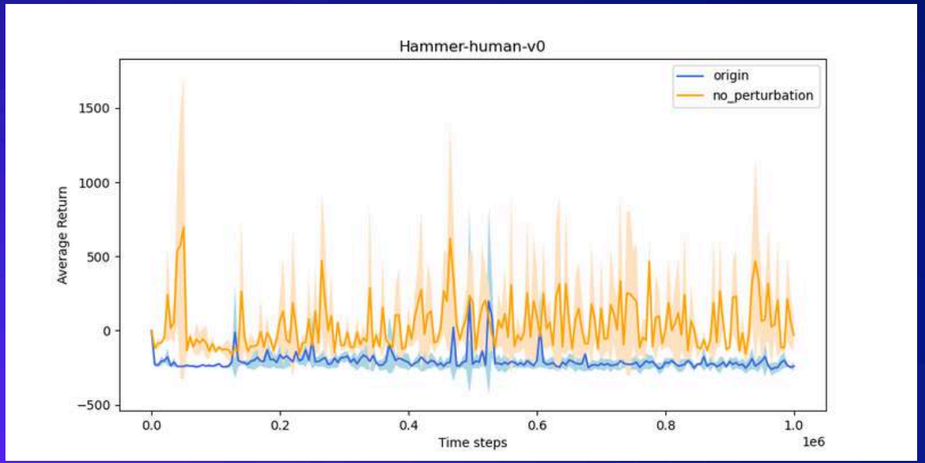


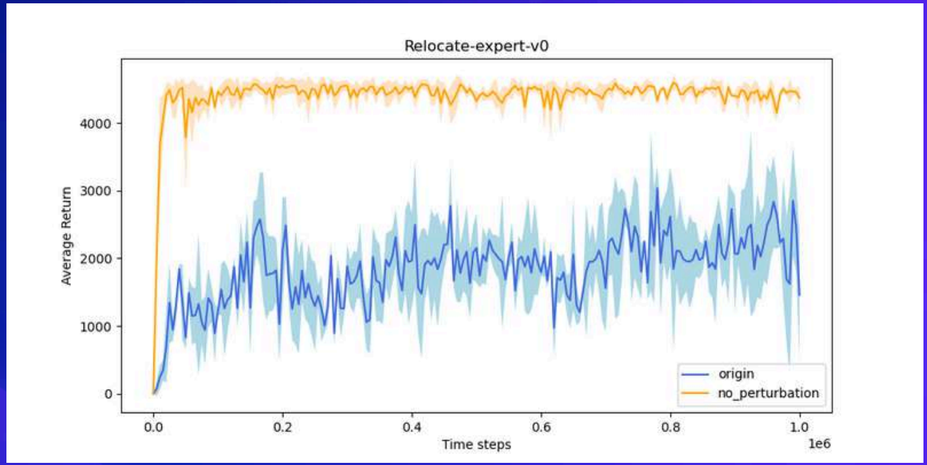
RESULTS OF STEP3

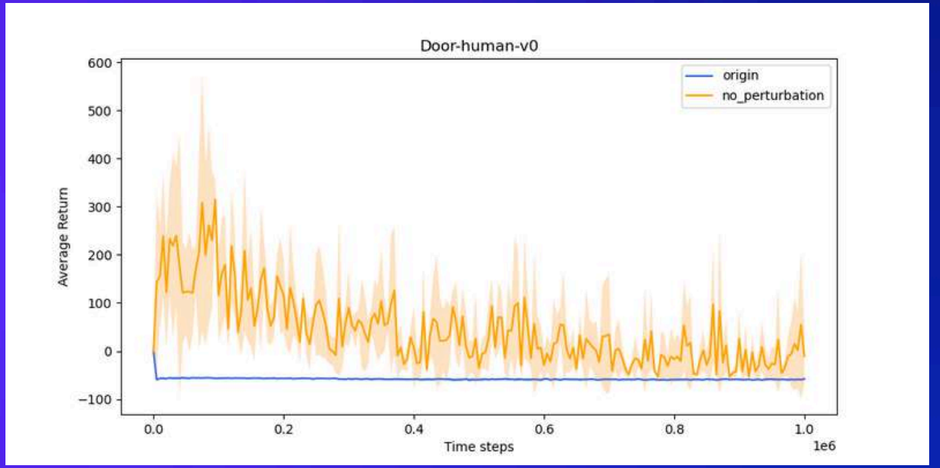
REMOVE PERTURBATION

It performs extremely well on complex tasks, especially on expert dataset. The reason is that it can more restrict to the state-action in the batch.





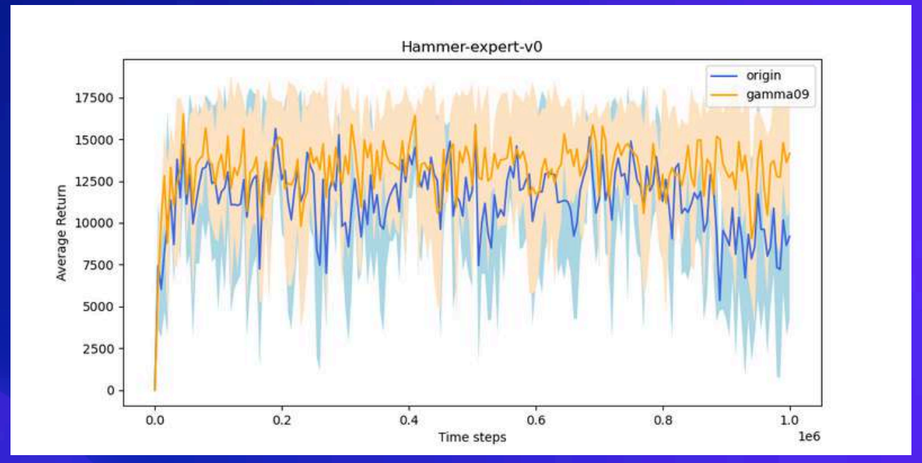


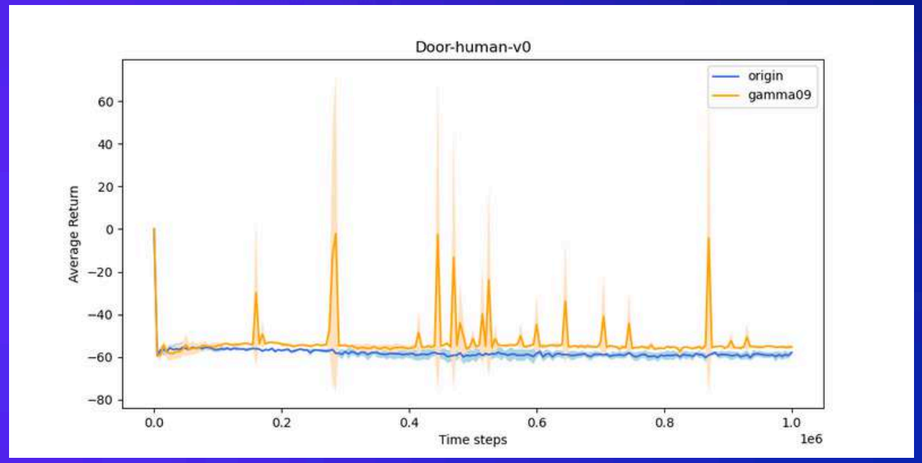


RESULTS OF STEP3

CHANGE DISCOUNT

Suprisingly, it perform well.

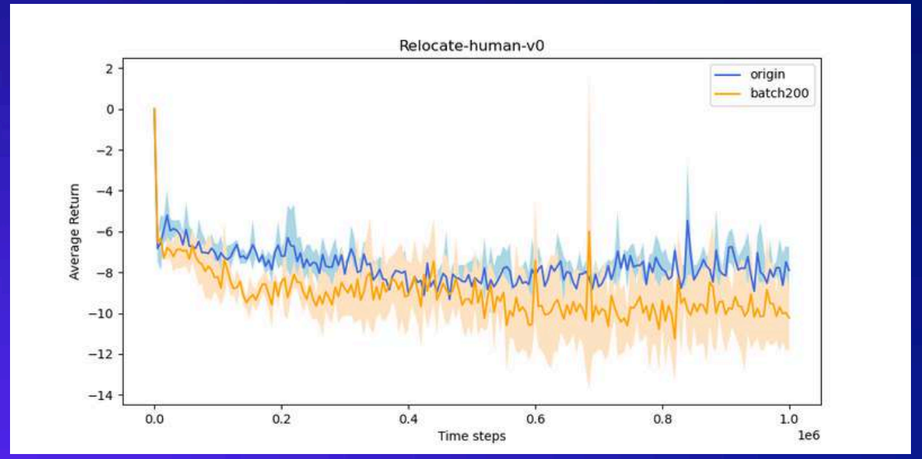


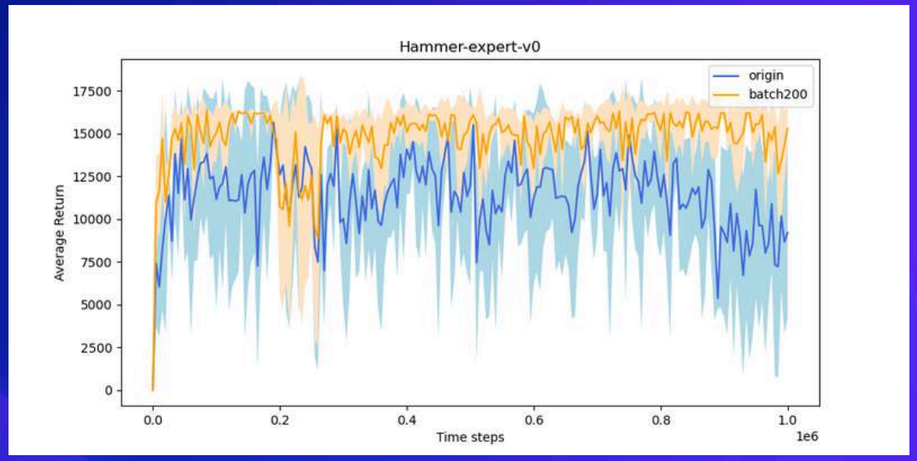


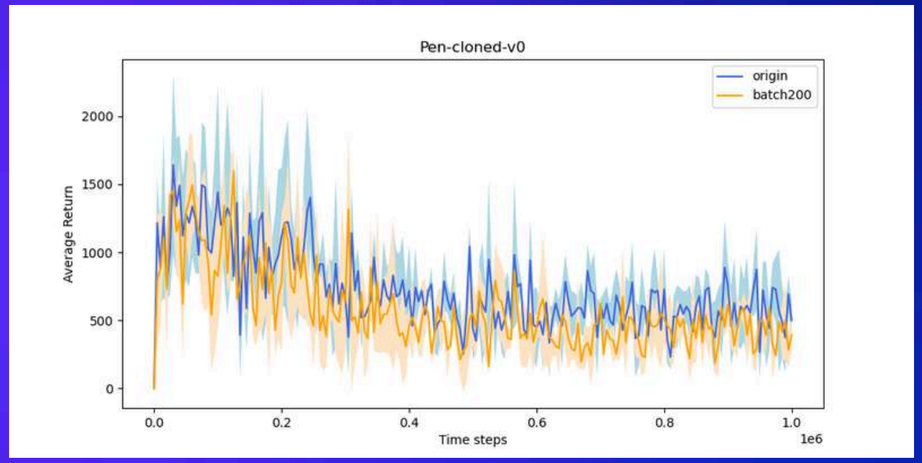
RESULTS OF STEP3

INCREASE BATCH SIZE

Sometimes model trained with larger batch size perform better than original one. Sometimes in opposite. It depend on the type of the task.







GITHUB REPOSITORY

RL_FINAL_PROJECT

REFERENCE

- https://arxiv.org/pdf/2004.07219
- https://arxiv.org/pdf/1812.02900
- https://arxiv.org/pdf/1802.09477v3
- https://github.com/sfujim/BCQ/tree/master
- https://blog.csdn.net/gsww404/article/details/123926753
- https://blog.csdn.net/weixin_43788106/article/details/123925913
- https://arxiv.org/pdf/1411.1784
- https://blog.csdn.net/yunlong_G/article/details/116375556
- https://blog.csdn.net/qq_24224067/article/details/104293409