Tree Chopping in Minecraft with Deep Q Learning from Demonstration

64340500005 Kittichet Arriyathanasak
64340500040 Punyawat Prachongkij

MineRL

MineRL is a Minecraft-based research environment designed for training and evaluating reinforcement learning agents with large dataset of human gameplay.

MineRLTreeChop_v0 Enviroment

MineRLTreeChop_v0 Enviroment is a Gym-compatible task where an agent learns to **chop trees** in Minecraft.

Reward & Observation

Terminated Condition
with 2000 step.

Obtain wood log,
agent gain reward

Visual gameplay are observation
(640 x 320 pixel with RGB colors)

Action List

1. Move forward (w key)
2. Turn camera with pitch = 0 degree and yaw = 5 degree
3. Attack (left mouse click)
4. Turn camera with pitch = 5 degree and yaw = 0 degree
5. Turn camera with pitch = -5 degree and yaw = 0 degree
6. Turn camera with pitch = 0 degree and yaw = -5 degree
7. Jump foward (w + space bar key)

Assumption

The demonstration of expert actions should guide the agent toward **optimal action** and **should help the algorithm perform better** than a pure RL-based Method

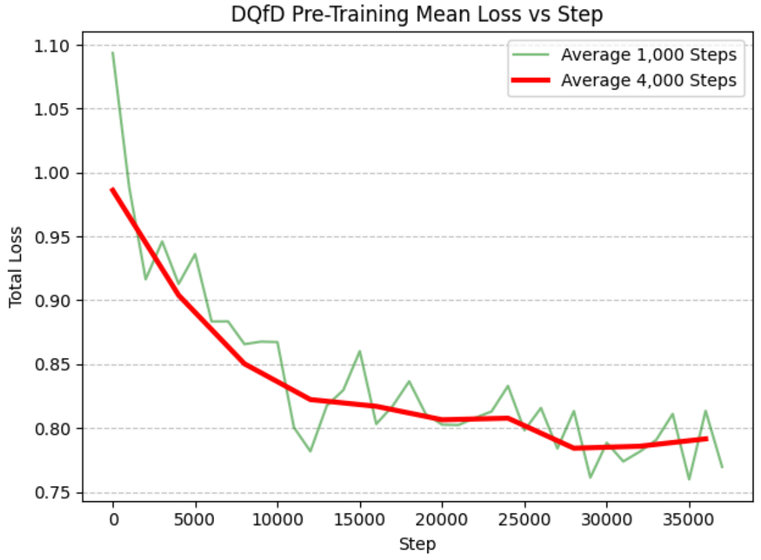
Experiment

Compare the algorithm with **Pure Imitation Learning** by the expert demonstration, **Pure Deep Reinforcement Learning** as a baseline **and Combine both method** as a implement algorithm

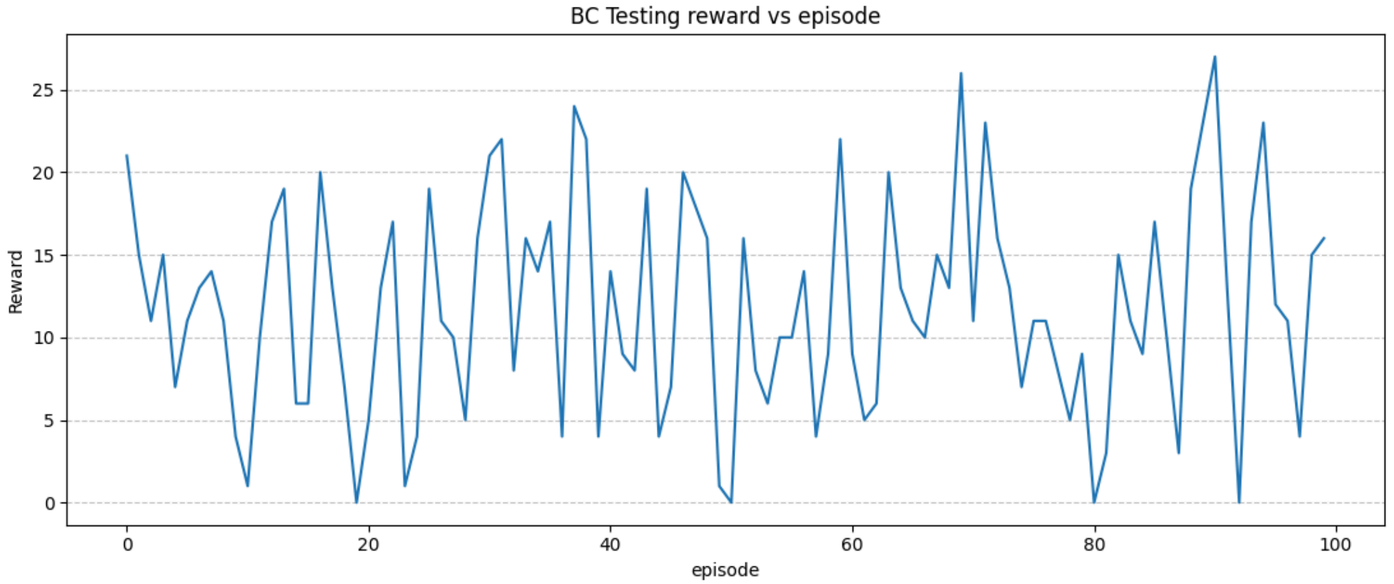
Behavior Cloning Result

Training model with CNN by supervised imitation learning

Cross Entropy Loss while Training



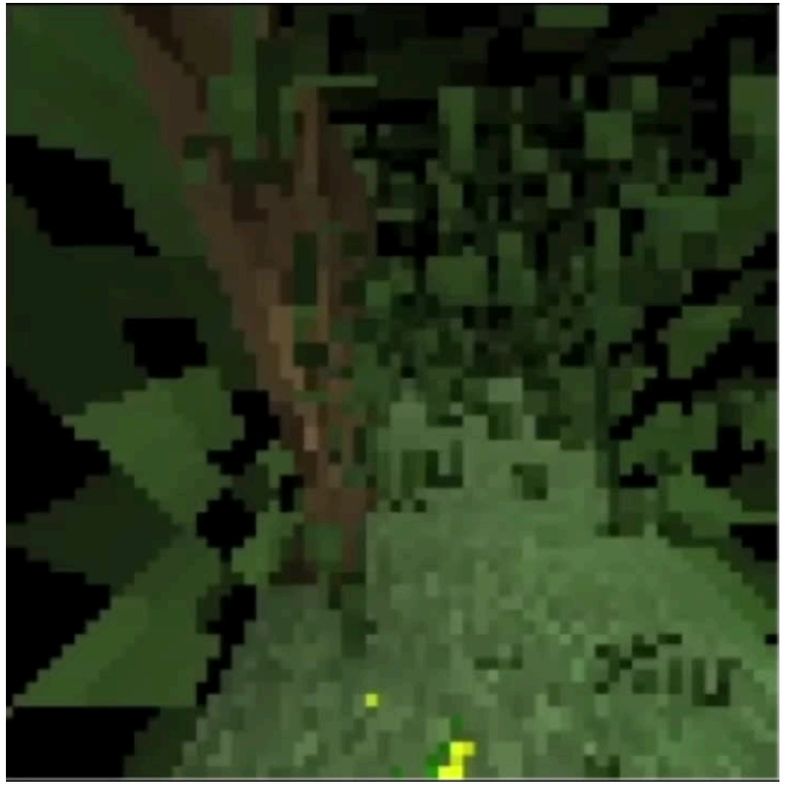
Testing reward with BC model



Average 100 Episode Reward : 11.79 +- 6.50

Behavior Cloning Result

Bad run



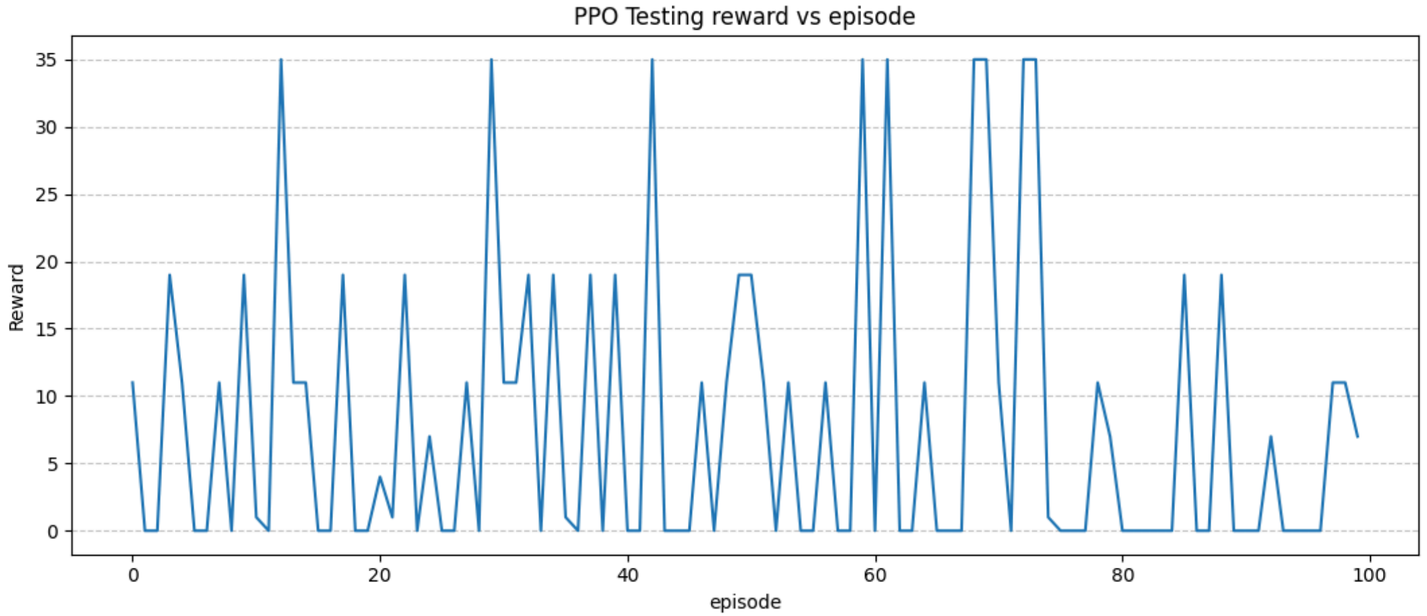
Good run

PPO Result

Training model with CNN by PPO with 1000 training episodes

PPO Loss while Training

Testing reward with PPO model



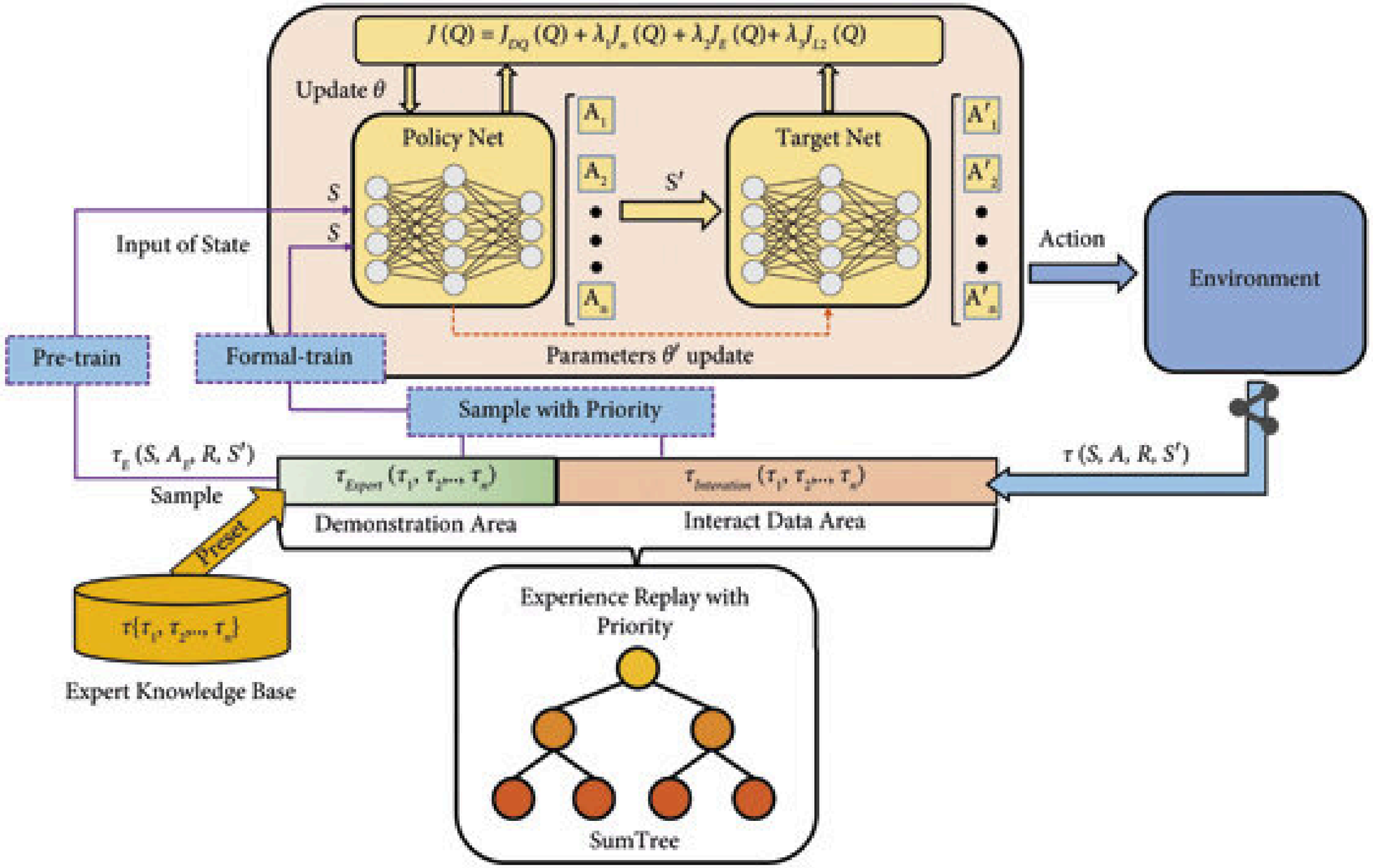
Average 100 Episode Reward : 7.77 +- 10.82

PPO Result

Bad run

Good run

Deep Q learning from Demonstration (DQfD)



Deep Q learning from Demonstration (DQfD)

While Pre-Train with Demonstration

$$J(Q)_s = J_Q(Q) + \lambda_1 J_n(Q) + \lambda_2 J_E(Q)$$

J = Total Loss
J_Q = TD Loss
J_n = n-Step TD Loss
J_E = Expert Loss
lamba_ 1, 2 = weight

Deep Q learning from Demonstration (DQfD)

While Train with Deep Q Learning

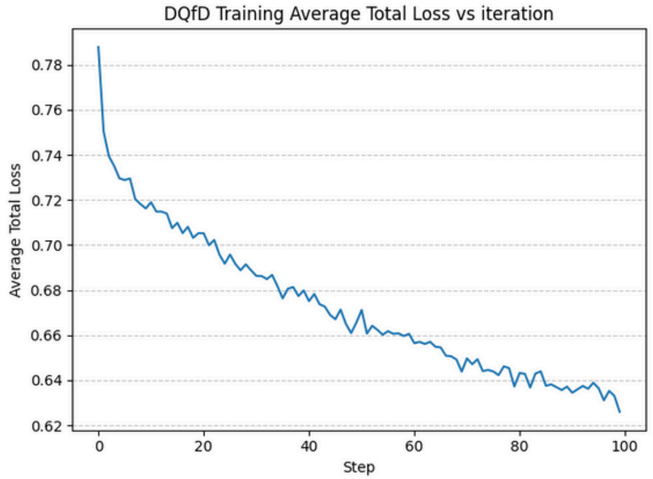
$$J(Q)_s = J_Q(Q) + \lambda_1 J_n(Q) + \lambda_2 J_E(Q)$$

0

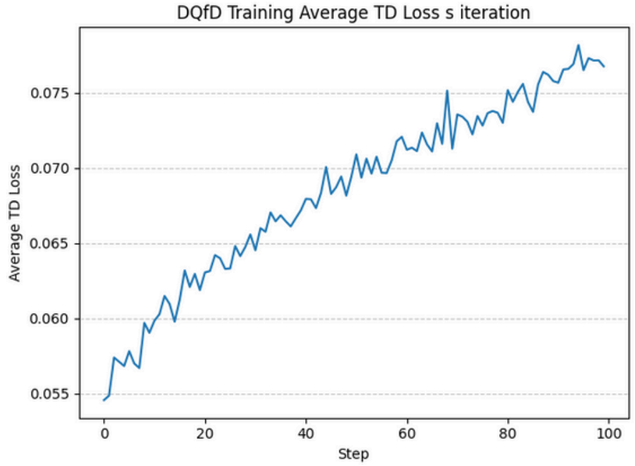
J = Total Loss
J_Q = TD Loss
J_n = n-Step TD Loss
J_E = Expert Loss
lamba_ 1, 2 = weight

Deep Q learning from Demonstration (DQfD)

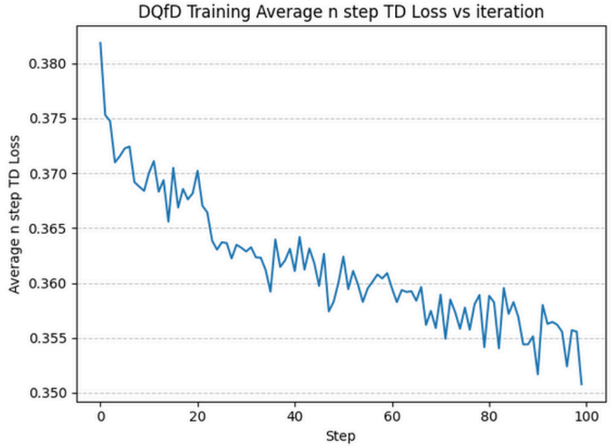
Pre-Train Loss



Total Loss



TD Loss

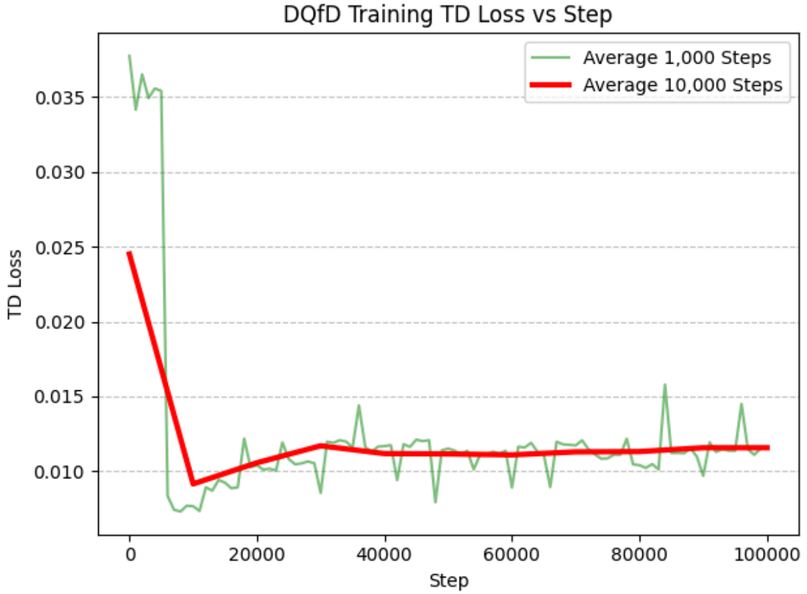


n-step TD Loss

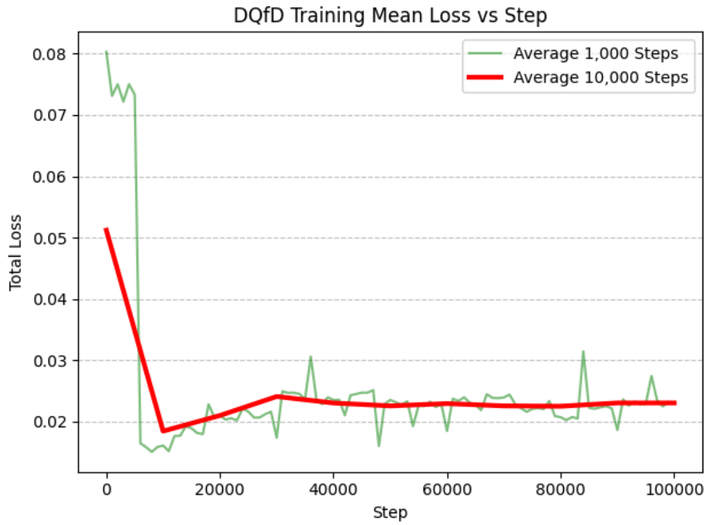
Expert Loss

Deep Q learning from Demonstration (DQfD)

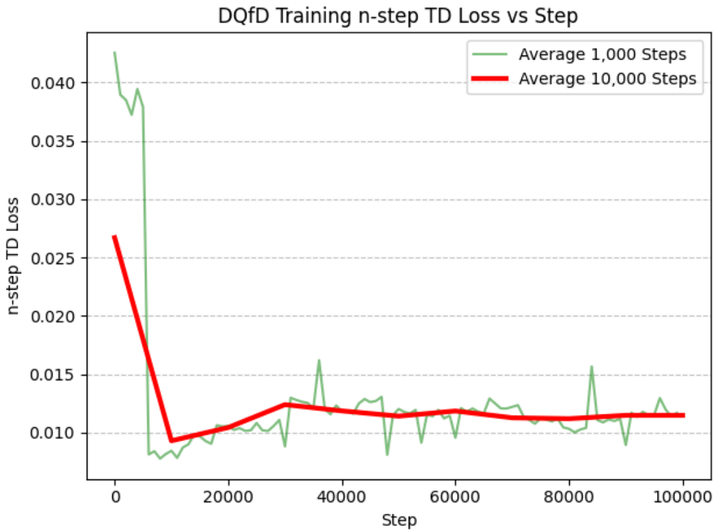
Training Loss



Total Loss

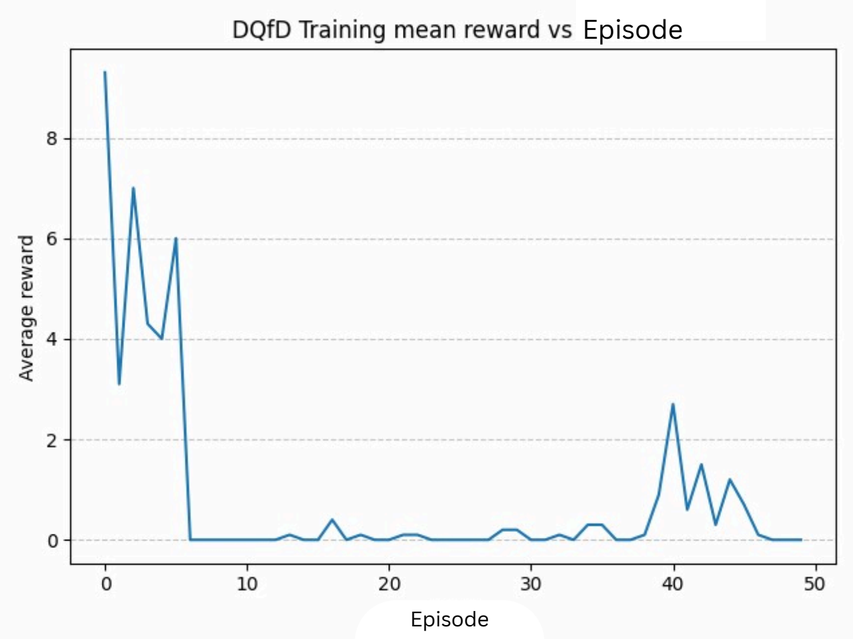


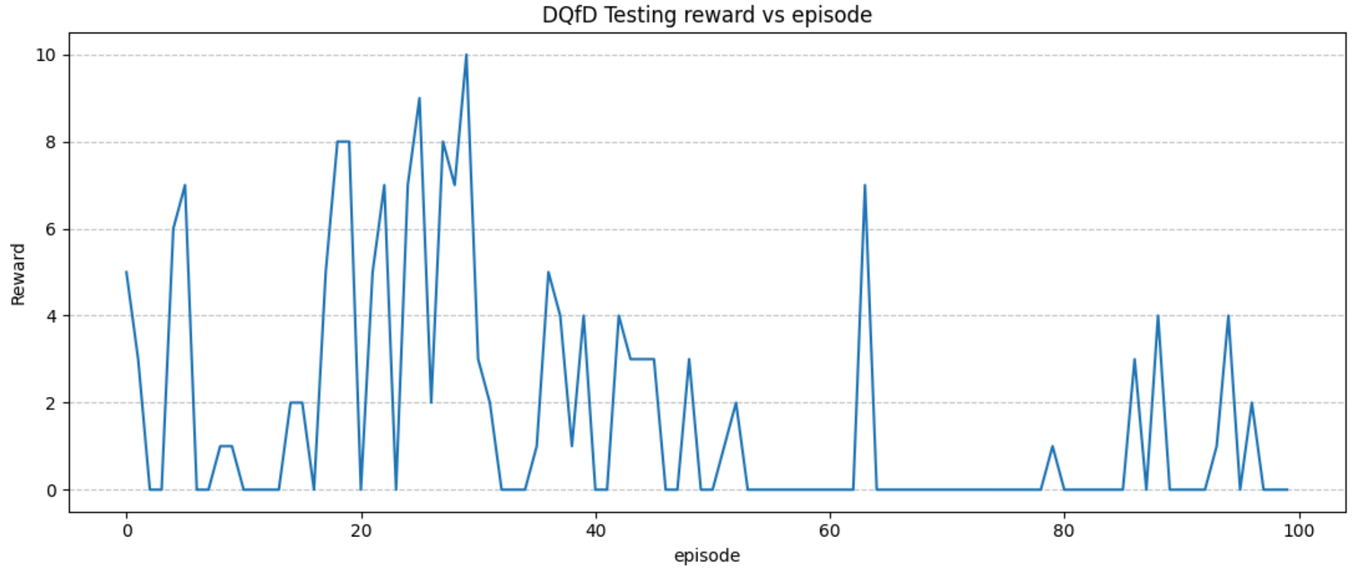
TD Loss



n-step TD Loss

Deep Q learning from Demonstration (DQfD)





Average 100 Episode Reward : 1.64 +- 2.56

DQfD Result

Experiment Result

Algorithm	Average Reward
BC	11.79 ± 6.50
PPO	7.77 ± 10.82
DQfD	1.64 ± 2.56

Conclusion

- The result of our implementation of DQfD is **not** as expected.
- The error in our implementation may stem from the **data sampling** method.

While Train with Deep Q Learning

$$J(Q)_s = J_Q(Q) + \lambda_1 J_n(Q) + \lambda_2 J_E(Q)$$

0

Conclusion

- The performance of PPO that is better than DQfD may stems from the **stochastic policy**.
- With proper **hyperparameter tuning**, RL-based method (PPO) might achieve better performance than BC.
- If we can formulate a **denser reward** (Distance to a nearest tree, time penalty), the RL-based method may generalize more and perform better in a bad seed.

Reference

1. Deep Q-learning from Demonstrations, Todd Hester et al., (AAAI 2018), available at https://arxiv.org/abs/1704.03732

2. SEIHAI: A Sample-efficient Hierarchical AI for the MineRL Competition, Hangyu Mao et al. (the third International Conference on Distributed Artificial Intelligence, 2018), available at https://arxiv.org/abs/2111.08857

3. Playing Minecraft with Behavioural Cloning, (NeurIPS2019, 2018), available at https://arxiv.org/abs/2005.03374