Supplementary Material: Backtrack Temporal Difference Learning-based Deep Reinforcement Learning for Sample-Efficient Robot Control Learning

I. Supplementary Material Overview

Section II describes the Backtrack DDPG (BT DDPG) algorithm. Section III supplements the experimental results. Section IV describes the network architectures and hyperparameters of BT DQN and BT DDPG.

II. Backtrack DDPG Algorithm

Algorithm 1 summarizes the BT DDPG algorithm.

Algorithm 1 Backtrack DDPG

Input:
θ, θ^-: the parameters of Q and target Q network
ω: the parameters of policy network
D: empty replay buffer
Parameter: β: learning rate; M: batch size; K: number of actors; η: the update parameter between Q network and its target network; N_m: replay buffer maximum size

Launch K actors and replicate network weights (θ, ω) to each actor
for traning_step $<$ total_training_step **do**
  Observe state s_t, and sample action a_t from policy
  Receive next state s_{t+1}, reward r_t from environment
  if $|D| > N_m$ **then**
    Delete the oldest sample (s, a, r, s')
  end if
  Add sample (s_t, a_t, r_t, s_{t+1}) to replay buffer D
  Sample a mini-batch samples from replay buffer D according to backtrack Eqs. (12) - (15)
  Compute the Backtrack TD-error loss: run Eq. (10)
  $\theta \leftarrow \theta - \beta \cdot \nabla_\theta \mathcal{L}^{TD}(\theta)$
  Update the policy network:
  $\delta_\omega = \frac{1}{M} \sum_m \nabla_\omega J(\mu_\omega)$
  $\omega \leftarrow \omega + \alpha_t \delta_\omega$
  if $l \bmod value\_update\_interval = 0$ **then**
    update target value function network:
    $\theta^- \leftarrow \eta\theta + (1-\eta)\theta^-$
  end if
end for

Output: critic and policy parameters θ, ω

III. Experimental Results

In Section III-A, we evaluated BT DQN on Robot Grid World Navigation task with truly state-action values to verify BT DQN's superior function approximations ability than baselines. In Section III-B, we describe the observation space, action space, and reward settings of CartPole and LunarLander tasks [1]. In Section III-C, we describe the observation space, action space, and reward settings of Reacher and BipedalWalker tasks [2].

A. Experiments of BT DQN on Robot Grid World Navigation

We evaluated BT DQN on the Robot Grid World Navigation task to verify the superior function approximation ability. We compared BT DQN with DQN and DQN with prioritized experience replay (PER DQN).

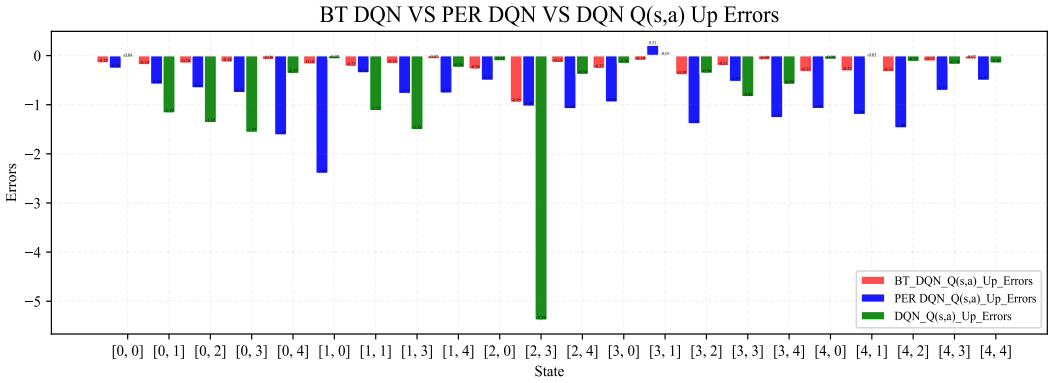
Fig. 1 (a) shows the *up* state-action pairs' $Q(s, a)$ values approximation errors of DQN, PER DQN, and BT DQN. Similarly, Fig. 1 (b) (c) (d) show the *up*, *left*, and *right* $Q(s, a)$ value approximation errors, respectively. We name these four classes of errors as Up_Errors, Down_Errors, Left_Errors, and Right_Errors. Fig. 1 demonstrates that BT DQN outperforms DQN and PER DQN on these four classes' errors.

B. Classic Robot Control Tasks

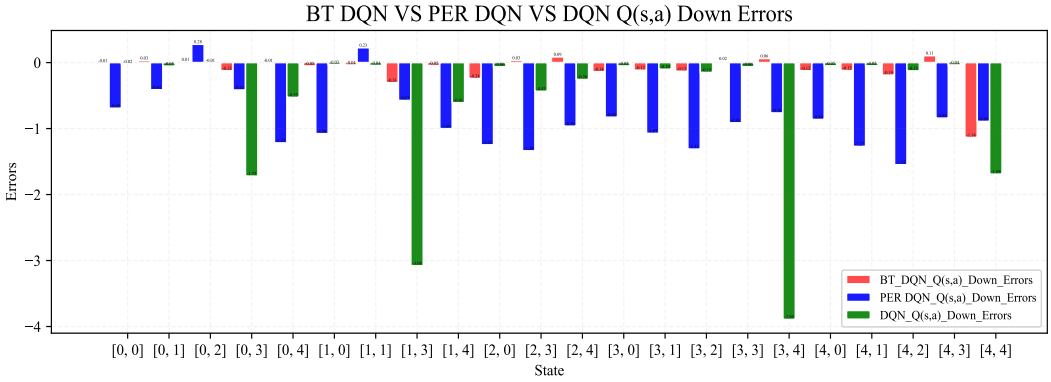
CartPole and LunarLander are representative and wildly used robot control benchmarks in deep RL. These classic robot control tasks aim to allow researchers to focus on the design of RL algorithms and experiments instead of environment design. The observation space, action space, and reward settings of CartPole and LunarLander tasks are as follows.

CartPole: The observation is a vector with shape 4 with the values corresponding to the following positions and velocities: cart position, cart velocity, pole angle, and pole angular velocity. The action is a vector with shape one which can take values 0, 1 indicating the direction of the force. The cart is pushed with: push the cart to the left and push the cart to the right. Because the goal is to keep the pole upright for as long as possible, a reward of +1 for every step taken, including the termination step, is allotted.

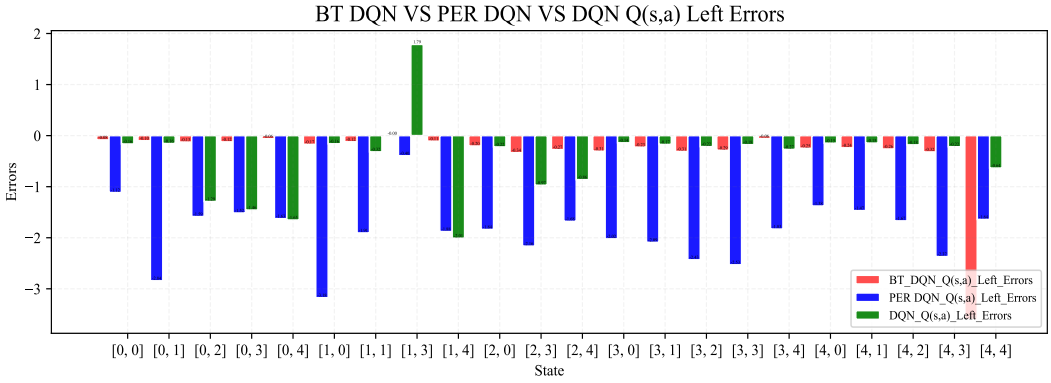
LunarLander: The state is an 8-dimensional vector: the coordinates of the lander in (x, y), its linear velocities in



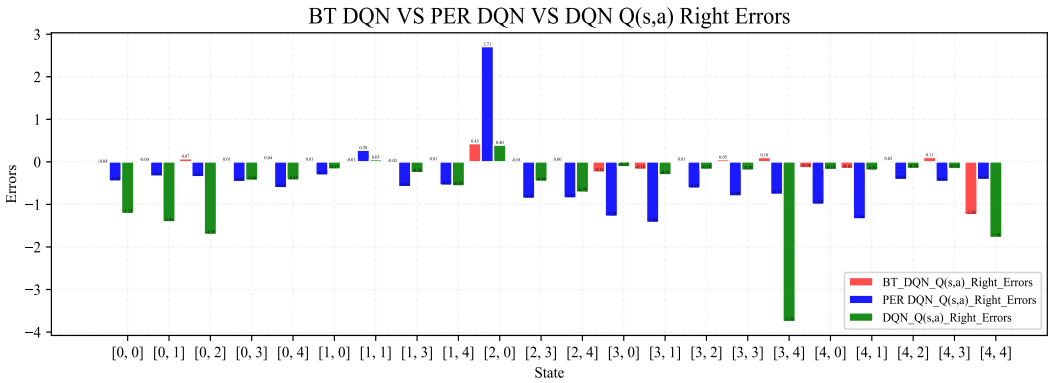
(a) $Q(s,a)$ Up_Errors of DQN, PER DQN and BT DQN



(b) $Q(s,a)$ Down_Errors of DQN, PER DQN and BT DQN



(c) $Q(s,a)$ Left_Errors of DQN, PER DQN and BT DQN



(d) $Q(s,a)$ Right_Errors of DQN, PER DQN and BT DQN

Fig. 1. $\boldsymbol{Q(s,a) - Q_*(s,a)}$ errors of DQN, PER DQN, and BT DQN.

(x, y), its angle, its angular velocity, and two booleans that represent whether each leg is in contact with the ground or not. Four discrete actions are available: do nothing, fire the left orientation engine, fire the main engine, and fire the right orientation engine. The reward for moving from the top of the screen to the landing pad and coming to rest is about 100-140 points. If the lander moves away from the landing pad, it loses the reward. If the lander crashes, it receives an additional -100 points. If it comes to rest, it receives an additional +100 points. Each leg with ground contact is +10 points. Firing the main engine is -0.3 points in each frame. Firing the side engine is -0.03 points in each frame. Solved is 200 points.

C. DM-Control tasks

The observation space, action space, and reward settings of Reacher and BipedalWalker tasks [2] are as follows.

Reacher: There is a two-link planar reacher with a randomly located target. The state space is a vector with shape 4, and the action space is a vector with shape 2. When the end effector enters the target sphere, the reward is +1.

BipedalWalker: The state consists of hull angle speed, angular velocity, horizontal speed, vertical speed, position and angular speed of joints, legs contact with the ground, and ten lidar rangefinder measurements. There are no coordinates in the state vector. Actions are motor speed values in the [-1, 1] range for each of the four joints at both hips and knees. The reward is given for moving forward, totaling 300+ points up to the far end. If the robot falls, it gets -100. Applying motor torque costs a small number of points. A more optimal agent will get a better score.

IV. Network Architectures and Hyperparameters

This section describes the network architectures and hyperparameters of BT DQN and BT DDPG.

A. Network architectures and hyperparameters of BT DQN

Table I lists the network architecture on Robot Grid World Navigation and Classic Robot Control experiments of BT DQN. Table II lists the network architecture on Robotic Grasping experiment of BT DQN. Table III lists the hyperparameters on Robot Grid World Navigation, Classic Robot Control, and Robotic Grasping experiments of BT DQN.

TABLE I
NETWORK ARCHITECTURE ON ROBOT GRID WORLD NAVIGATION AND CLASSIC ROBOT CONTROL EXPERIMENTS OF BT DQN

Layers	Architecture
Layer1	ReLU(Linear(state_dim, 256))
Layer2	ReLU(Linear(256, 256))
Layer3	ReLU(Linear(256, action_dim))

TABLE II
NETWORK ARCHITECTURE ON ROBOTIC GRASPING EXPERIMENT OF BT DQN

Layers	Architecture
Layer1	ReLU(BatchNorm2d(Conv2d(state_dim, 32, kernel_size=8, stride=4)))
Layer2	ReLU(BatchNorm2d(Conv2d(32, 64, kernel_size=4, stride=2)))
Layer3	ReLU(BatchNorm2d(Conv2d(64, 64, kernel_size=3, stride=1)))
Layer4	ReLU(Linear(linear_input_size, 512))
Layer5	ReLU(Linear(512, action_dim))

B. Network architectures and hyperparameters of BT DDPG

Table IV lists the network architecture on the Robot 2D task experiment of BT DDPG. Table V lists the network architecture on DM-Control experiments of BT DDPG. Table VI lists the network architecture on the Robotic Grasping experiment of BT DDPG. Table VII lists the Image Feature Extraction Architecture on Robotic Grasping experiment of BT DDPG. Table VIII lists the hyperparameters on Robot 2D task, DM-Control, and Robotic Grasping experiments of BT DDPG.

TABLE III
HYPERPARAMETERS ON GRID WORLD, CLASSIC ROBOT CONTROL, AND ROBOTIC GRASPING EXPERIMENTS OF BT DQN

Hyperparameters	Grid World	Classic Robot Control	Robotic Grasping
discount factor	0.9	0.99	0.9
training epoch	7000	3000	5000
learning rate	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$
replay buffer size	$1e^{4}$	$1e^{4}$	$1e^{4}$
batch size	64	64	32
ϵ-greedy	$1-\frac{current\_step}{total\_step}$	$1-\frac{current\_step}{total\_step}$	$1-\frac{current\_step}{total\_step}$

TABLE IV
NETWORK ARCHITECTURE ON ROBOT 2D TASK EXPERIMENT OF BT DDPG

Layers	Actor	Critic
Layer1	ReLU(Linear(state_dim, 40))	ReLU(Linear(state_dim+action_dim, 40))
Layer2	ReLU(Linear(40, 30))	ReLU(Linear(40, 30))
Layer3	Tanh(Linear(30, action_dim))	ReLU(Linear(30, 1))

TABLE V
NETWORK ARCHITECTURE ON DM-CONTROL EXPERIMENT OF BT DDPG

Layers	Actor	Critic
Layer1	ReLU(Linear(state_dim, 256))	ReLU(Linear(state_dim+action_dim, 256))
Layer2	ReLU(Linear(256, 256))	ReLU(Linear(256, 256))
Layer3	Tanh(Linear(256, action_dim))	ReLU(Linear(256, 1))

TABLE VI
NETWORK ARCHITECTURE ON ROBOTIC GRASPING EXPERIMENT OF BT DDPG

Layers	Actor	Critic
Layer1	Image Feature Extraction Architecture (Table VII)	Image Feature Extraction Architecture (Table VII)
Layer2	ReLU(Linear(linear_input_size, 256))	ReLU(Linear(linear_input_size+action_dim, 256))
Layer3	ReLU(Linear(256, 256))	ReLU(Linear(256, 256))
Layer4	Tanh(Linear(256, action_dim))	ReLU(Linear(256, 1))

TABLE VII
IMAGE FEATURE EXTRACTION ARCHITECTURE ON ROBOTIC GRASPING EXPERIMENT OF BT DDPG

Layers	Architecture
Layer1	ReLU(BatchNorm2d(Conv2d(state_dim, 32, kernel_size=8, stride=4)))
Layer2	ReLU(BatchNorm2d(Conv2d(32, 64, kernel_size=4, stride=2)))
Layer3	ReLU(BatchNorm2d(Conv2d(64, 64, kernel_size=3, stride=1)))

TABLE VIII
HYPERPARAMETERS ON ROBOT 2D TASK, DM-CONTROL, AND ROBOTIC GRASPING EXPERIMENTS OF BT DDPG

Hyperparameters	Robot 2D	DM-Control	Robotic Grasping
discount factor	0.9	0.99	0.99
training epoch	3000	3000	5000
learning rate	$1e^{-4}$	$1e^{-3}$	$1e^{-3}$
replay buffer size	$5e^{3}$	$1e^{6}$	$1e^{6}$
batch size	64	100	100

REFERENCES

[1] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[2] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa, "dm_control: Software and tasks for continuous control," *Software Impacts*, vol. 6, p. 100022, 2020.