Enhanced mobile robot navigation using TD3 and DQN

Tathya Bhatt | Modabbir Adeeb
120340246 120314827

01 # Overview

Introduction
TD3 and DQN

02 # Methodology

Workflow
Parameters

03 # Results & Improvements

Comparison of Results
Future Scope

04 # References

Repos and Paper

Introduction

This project focuses on the implementation of deep reinforcement learning algorithms on a mobile robot to navigate a given environment with obstacles using two algorithms: TD3 and DQN.

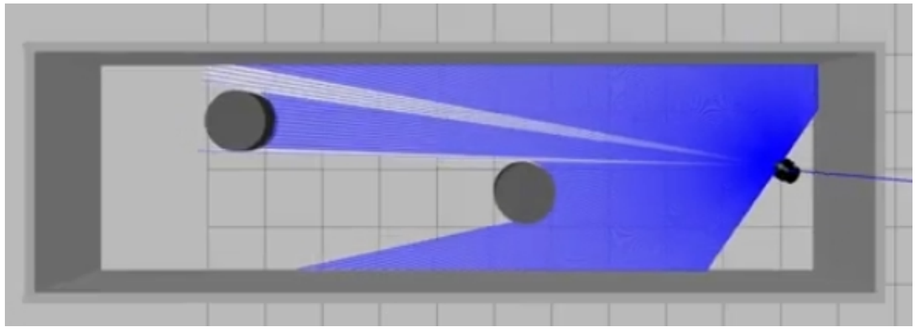
Optimized hyper-parameters to improve the convergence of Q-values.

Results are compared for two architectures:

- Shallow Neural Network
- Deep Neural Network

Tools Used

- ROS2 Humble
- Gazebo
- RViz
- Velodyne LiDAR



Gazebo Environment with LiDAR Rays on Mobile Robot

Twin Delayed Deep Deterministic Policy Gradient (TD3)

- Combines aspects of DDPG with improvements to address function approximation errors
- Actor-Critic Architecture
  - Actor network optimizes the policy.
  - Critic network evaluates action-value pairs.
- Training: Utilizes experience replay for learning from past experiences.
- Suited for environments with continuous action spaces
- Application: Used in robotic control, autonomous driving, and other applications requiring precise continuous control.

Algorithm 1 TD3

1 Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$

2 Initialize replay buffer $\mathcal{B}$

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

3 Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'

4 Store transition tuple (s, a, r, s') in $\mathcal{B}$

5 Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
$\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
$y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$

6 **if** t mod d **then**
Update ϕ by the deterministic policy gradient:
$\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$

7 Update target networks:
$\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
$\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

end if

end for

https://towardsdatascience.com/td3-learning-to-run-with-ai-40dfc512f93

Deep Q-Learning (DQN)

- Reinforcement learning algorithm combining Q-learning with deep neural networks.
- Training:
  - Employs experience replay, a technique that stores and randomly samples past experiences to break temporal correlations, enhancing learning stability.
- To stabilize training, DQN uses two sets of networks—one for determining actions (policy network) and another for evaluating Q-values (target network).
- Application: Effective in playing video games, robotics, and other domains requiring decision-making in large state spaces.

Algorithm 1 TD3

1 Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$

2 Initialize replay buffer $\mathcal{B}$

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

3 Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'

4 Store transition tuple (s, a, r, s') in $\mathcal{B}$

5 Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
$\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
$y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$

6 **if** t mod d **then**
Update ϕ by the deterministic policy gradient:
$\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$

7 Update target networks:
$\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
$\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$

end if

end for

Workflow

-TD3

- Created Gazebo world and deployed mobile robot
- Implemented the algorithm in a custom-made environment
- Addition of an extra layer of the neural network
- Using ROS Odometry information to calculate current position
- Comprehensive analysis of Shallow and Deep Neural Network to evaluate the performance using:
  -Average Q-values
  -Maximum Q-value
  -Network Loss

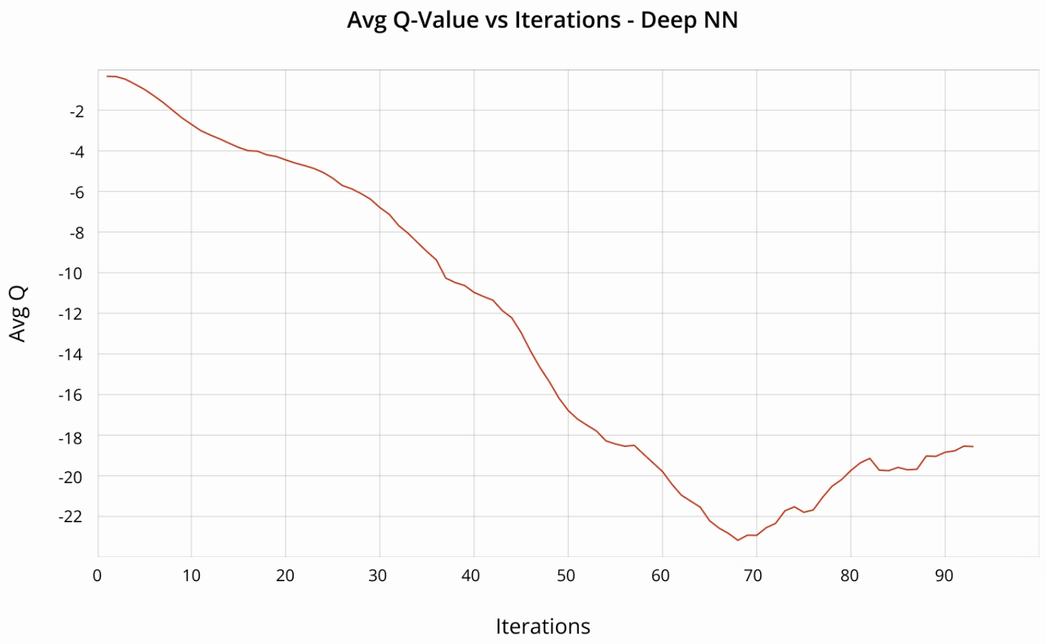
-DQN

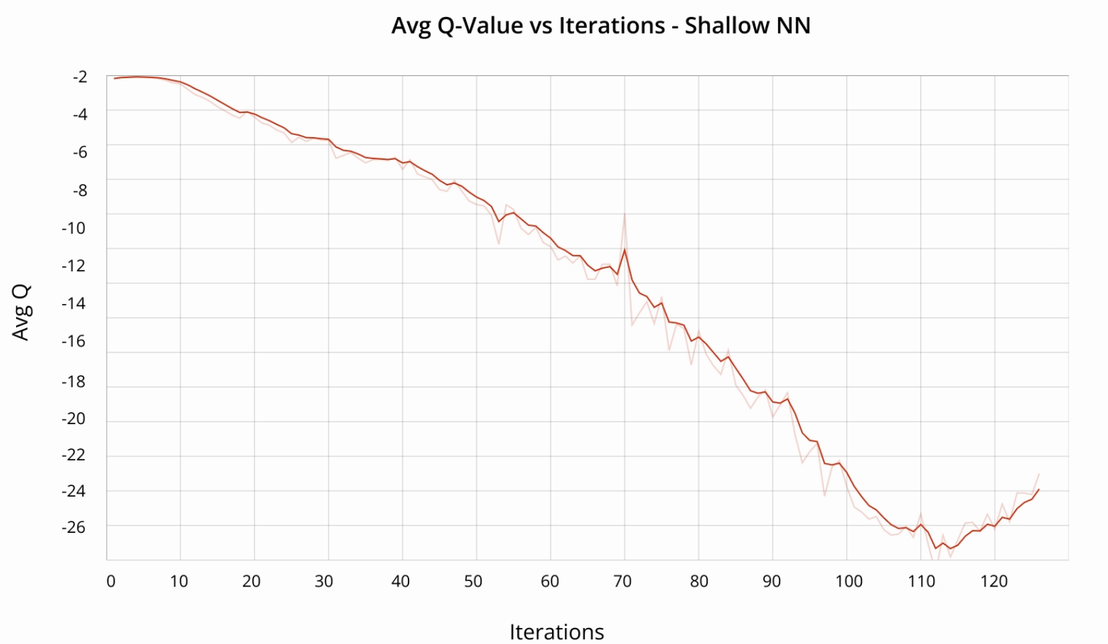
- Implementation of DQN algorithm in code to represent the maximum Q-values obtained.

Training Parameters

- Coefficient for updating the target network: 0.005
- Discount Factor: 0.99
- Noise Clip Threshold: 0.5
- Noise for exploration: 0.2
- Batch size: 100
- Buffer size: 1e6
- Epoch: 1
- Maximum Timesteps: 3e6
- Maximum episodes: 300
- Optimizer: Adam
- Evaluation frequency: 5000
- Evaluation Episodes: 10
- Training time: 45 minutes

Results

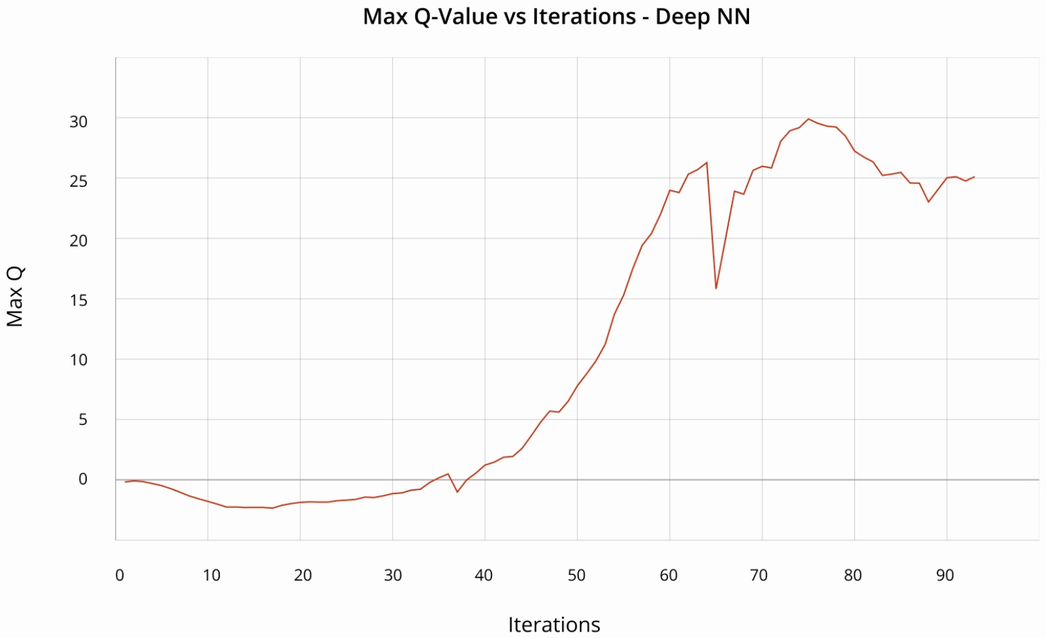


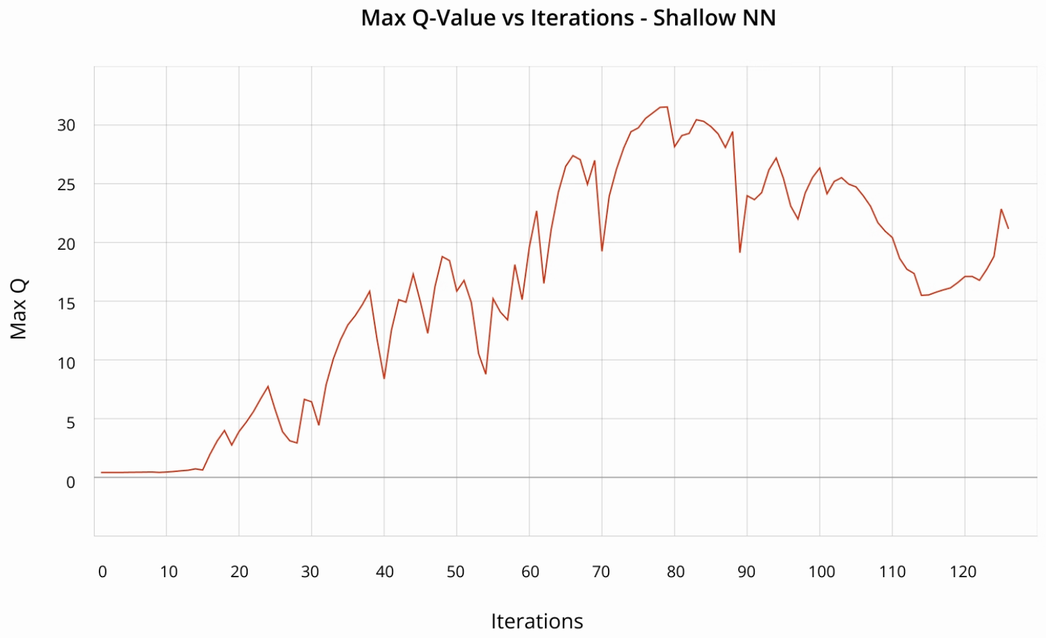


- More faster reduction in deep neural network
- Less Iterations
- Computationally Expensive

Avg Q-Value

Results

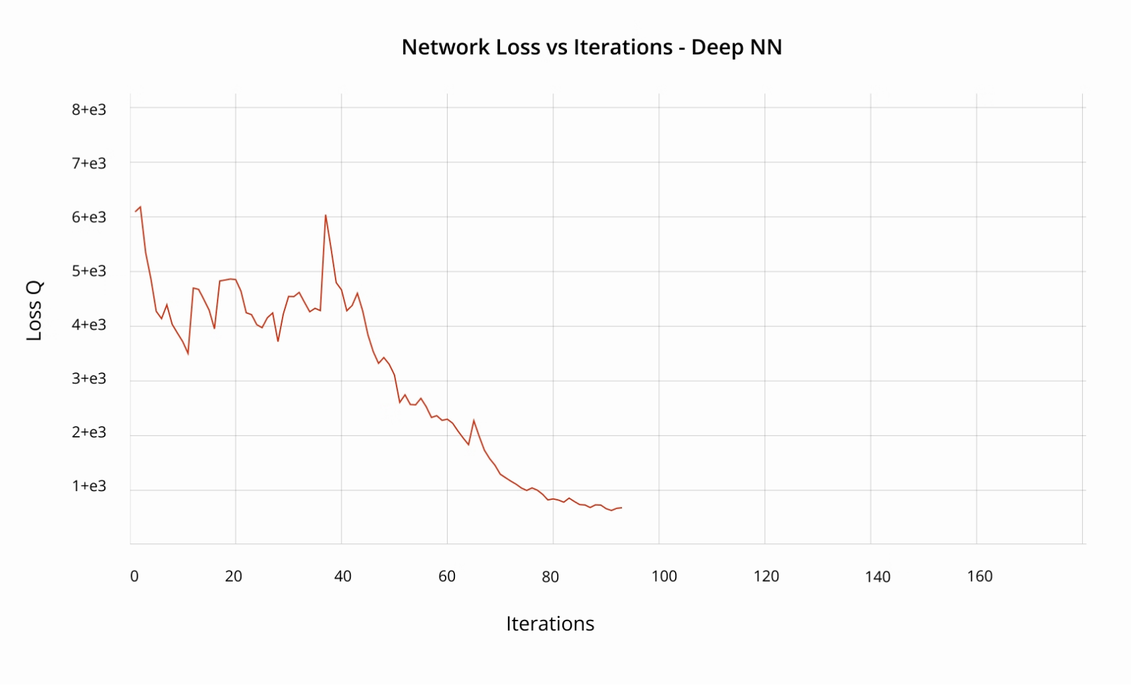


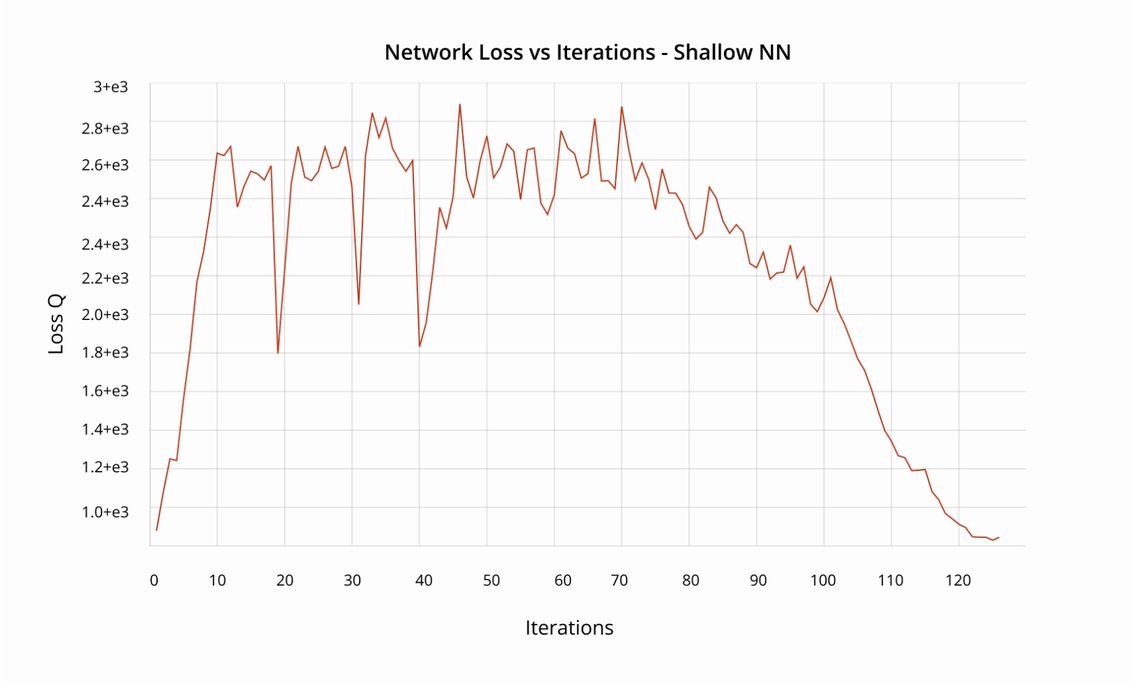


- Stable change in q values
- Less Iterations

Max Q-Value

Results

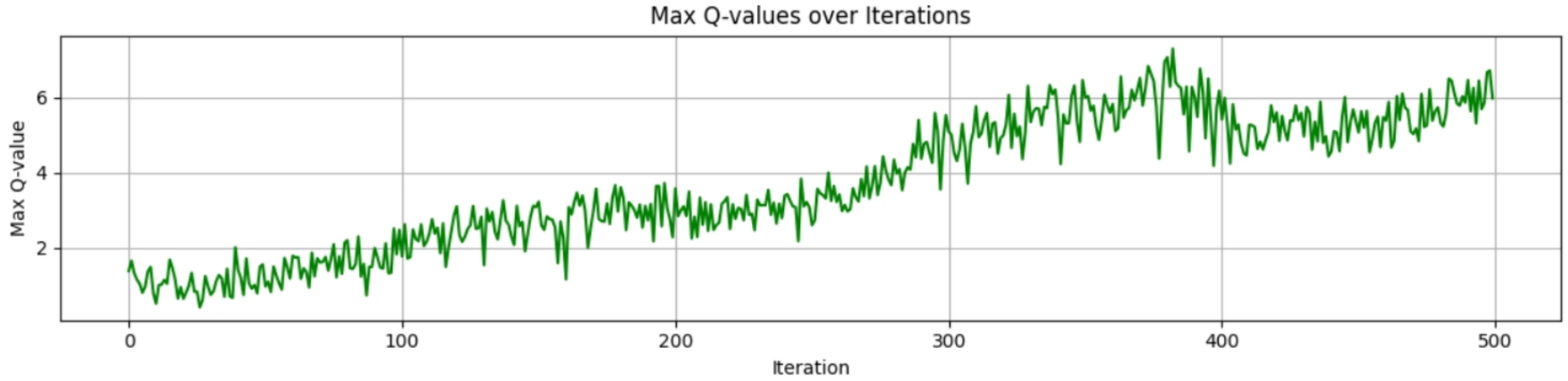




- More faster reduction in network loss
- Higher initial loss

Q - Loss

Results



Basic DQN

Demo

Training & Validation

https://drive.google.com/file/d/1czt7jeloPETLZ584lH3S5WX08z9aTNkd/view?usp=drive_link

Testing

https://drive.google.com/file/d/1czt7jeloPETLZ584lH3S5WX08z9aTNkd/view?usp=drive_link

Improvements

- Limiting the state-action generation to a constrained space instead of learning multiple state-action pair
- Increase in the number of iteration to better train the model
- Accuracy can be increased with increasing epochs and episodes

- Stabilizing the robot and remove wobbling while braking

Demo

References

1. I. S. Peyas, Z. Hasan, M. R. Rahman Tushar, A. Musabbir, R. M. Azni and S. Siddique, "Autonomous Warehouse Robot using Deep Q-Learning," TENCON 2021 - 2021 IEEE Region 10 Conference (TENCON), Auckland, New Zealand, 2021, pp. 857-862, doi: 10.1109/TENCON54134.2021.9707256.

2. https://medium.com/@reinis_86651/deep-reinforcement-learning-in-mobile-robot-navigation-tutorial-part3-training-13b2875c7b51

3. https://github.com/reiniscimurs/DRL-robot-navigation

4. Jeng SL, Chiang C. End-to-End Autonomous Navigation Based on Deep Reinforcement Learning with a Survival Penalty Function. Sensors (Basel). 2023 Oct 23;23(20):8651. doi: 10.3390/s23208651. PMID: 37896743; PMCID: PMC10610759.

5. https://github.com/zerosansan/dqn_qlearning_sarsa_mobile_robot_navigation

6. https://github.com/sfujim/TD3

7. R. Cimurs, I. H. Suh and J. H. Lee, "Goal-Driven Autonomous Exploration Through Deep Reinforcement Learning," in IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 730-737, April 2022, doi: 10.1109/LRA.2021.3133591