Autonomous Vehicle Navigation using Reinforcement Learning

Course Name - ENPM 690 - Robot Learning
Instructor - Prof. Jerry Wu
Teaching Assistants - Aman Sharma, Sarin Mathew
Date - 08/05/24

Presenters : Joseph Thomas
Rishikesh Jadhav

Objectives

- To compare the performance of TD3 and DQN algorithms in terms of navigation efficiency, robustness, and computational requirements. This comparison will involve metrics such as navigation speed, efficiency and path smoothness.
- To investigate the ability of the algorithms to adapt to dynamic environments with changing obstacles.

Frameworks and Tools:

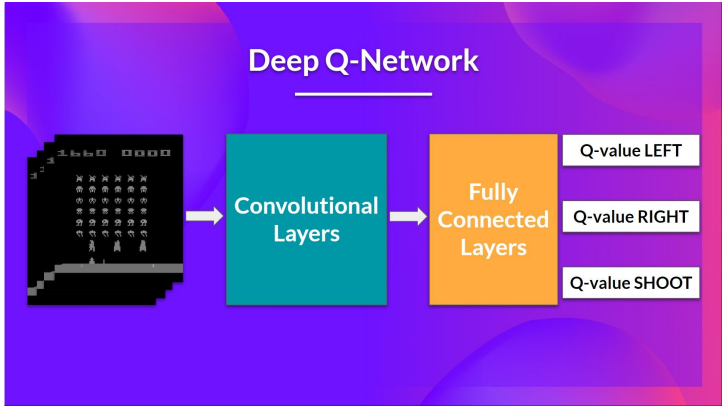
1. **Pytorch:** For implementing and training deep reinforcement learning algorithms such as TD3 and DQN.
2. **ROS2:** For robot control, communication, and integration with simulation environment.
3. **TensorBoard:** For visualization and analysis of training progress, performance metrics, and model evaluation.
4. **Gazebo:** Simulation Environment
5. **RViz:** 3D visualization

Algorithm 1 - Deep Q Networks

DQN is an algorithm designed to solve environments with discrete action spaces. It combines Q-learning, a traditional reinforcement learning algorithm, with deep neural networks. This integration enables the handling of high-dimensional, continuous state spaces, which are common in many real-world problems.

Advantages:

- DQNs can handle high dimensional input spaces, directly processing raw sensory inputs (like images) and learning control strategies, making them suitable for a wide range of applications, from video games to robotics.
- Deep neural networks can also generalize from seen to unseen states to some extent, which is crucial in complex environments where it's impractical to visit every possible state during training.



DQN Model Configuration

PARAMETER	VALUE
Device	cuda
Simulation Speed	1
State Size	44
Action Size	5
Hidden Size	512
Input Size	44
Batch Size	128
Buffer Size	1000000
Discount Factor	0.99
Learning Rate	0.003

PARAMETER	VALUE
Tau	0.003
Step Time	0.01
Loss Function	smooth_l1_loss
Epsilon	1.0
Epsilon Decay	0.9995
Epsilon Minimum	0.05
Reward Function	A
Backward Enabled	False

PARAMETER	VALUE
Stacking Enabled	False
Stack Depth	3
Frame Skip	4

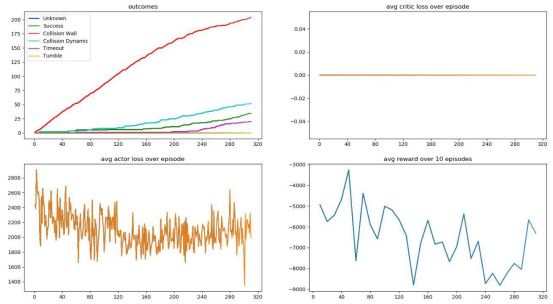
Networks:
- Actor 1
- Actor 2

Possible Actions:
- [0.3, -1.0]
- [0.3, -0.5]
- [1.0, 0.0]
- [0.3, 0.5]
- [0.3, 1.0]

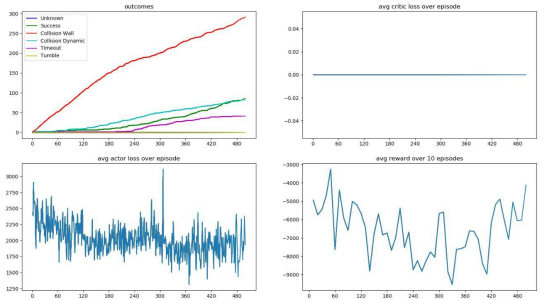
Actor Optimizer Details:
- amsgrad: False
- betas: (0.9, 0.999)
- eps: 1e-08
- lr: 0.003
- weight_decay: 0.01

DQN Testing Performance

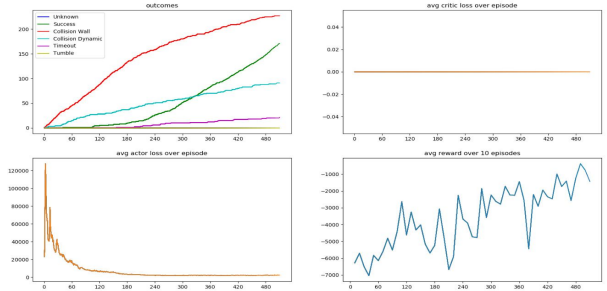
Performance without Hyperparameter Tuning at 300 epochs: Video Link



Performance without Hyperparameter Tuning at 500 epochs: Video Link



Performance with Hyperparameter Tuning at 500 epochs: Video Link

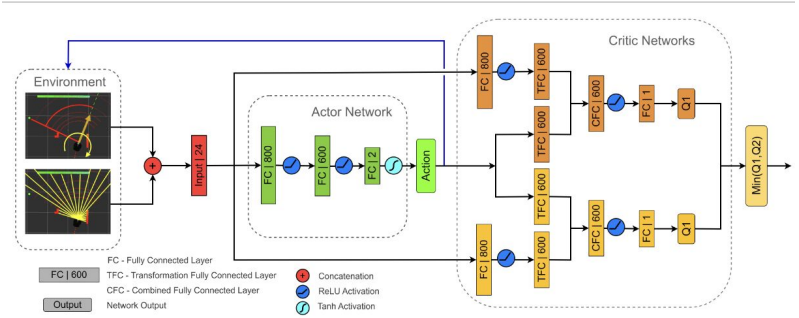


Algorithm 2 - Twin Delayed Deep Deterministic Policy Gradient (TD3)

- TD3 is an advanced algorithm designed for environments with continuous action spaces, expanding upon the Deep Deterministic Policy Gradient (DDPG) approach.
- It enhances stability and performance by utilizing twin Q-networks to mitigate overestimation bias, implementing delayed policy updates, and incorporating target policy smoothing.
- Following the actor-critic framework, TD3 employs a policy network ("actor") to select actions based on states and evaluates expected returns ("critic") for those actions.

Advantages:

- Superior performance in continuous action environments compared to DDPG and other policy gradient methods, making it ideal for complex control tasks and robotic applications.
- TD3's use of target policy smoothing and off-policy learning enables efficient exploration of the environment and effective policy learning in challenging scenarios.



TD3 Model Configuration

PARAMETER	VALUE
Device	cuda
Simulation Speed	1
State Size	44
Action Size	2
Hidden Size	512
Input Size	44
Batch Size	128
Buffer Size	1000000
Discount Factor	0.99
Learning Rate	0.003

PARAMETER	VALUE
Tau	0.003
Step Time	0.01
Loss Function	smooth_l1_loss
Epsilon	1.0
Epsilon Decay	0.9995
Epsilon Minimum	0.05
Reward Function	A
Backward Enabled	False

PARAMETER	VALUE
Stacking Enabled	False
Stack Depth	3
Frame Skip	4

Networks:
- Actors:
  - Actor 1
  - Actor 2
- Critics:
  - Critic 1
  - Critic 2

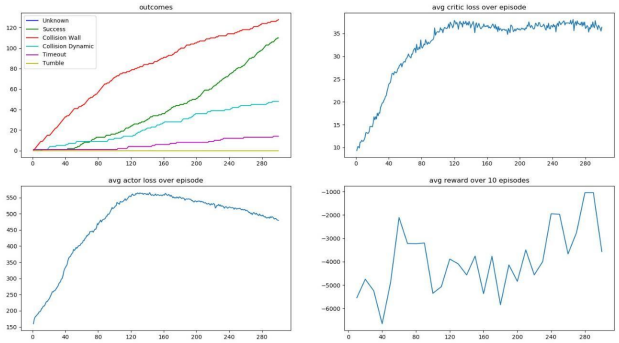
Actor Optimizer Details:
- amsgrad: False
- betas: (0.9, 0.999)
- eps: 1e-08
- lr: 0.003
- weight_decay: 0.01

Critic Optimizer Details:
- amsgrad: False
- betas: (0.9, 0.999)
- eps: 1e-08
- lr: 0.003
- weight_decay: 0.01

Other Details:
- Iteration: 0
- Noise: OUNoise object
- Policy Noise: 0.2
- Noise Clip: 0.5
- Policy Frequency: 2
- Last Actor Loss: 0

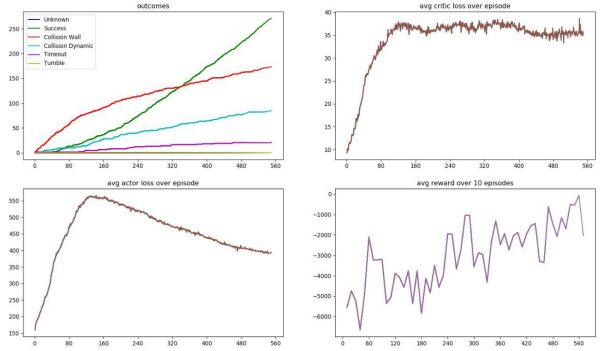
TD3 Testing Performance

Performance without Hyperparameter Tuning at 300 epochs:



Video Link

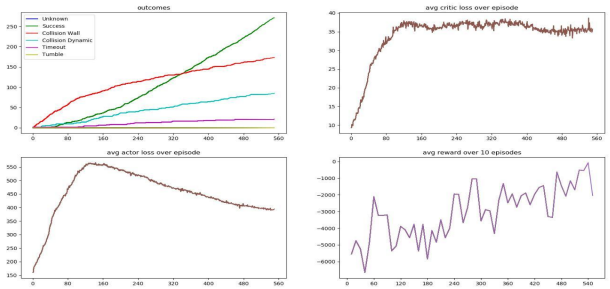
Performance with Hyperparameter Tuning at 500 epochs:



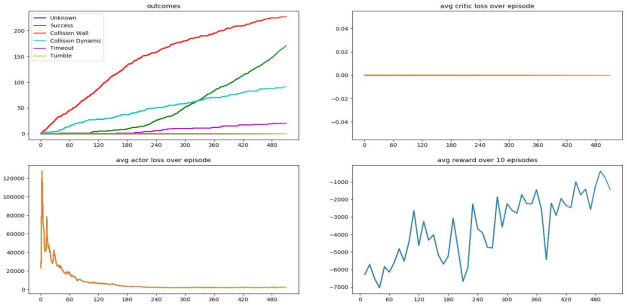
Video Link

Comparative Analysis and Conclusion: TD3 vs DQN

TD3 performance with hyperparameter tuning at 500 epochs: Video Link



DQN Performance with Hyperparameter Tuning at 500 epochs: Video Link



Conclusion :
The above graphs help summarize key metrics like average reward or navigation efficiency for both algorithms after hyperparameter tuning.

This helped us understand that DQN might be simpler to implement but TD3 algorithm achieved better results and offered better stability for continuous actions.

Limitations and Future Improvements:
Limitations include limited training time and specific simulation environment.

Future Improvements include exploring more advanced RL algorithms, incorporating real-world sensor data and testing in more complex environments.

References and Output Links

- https://arxiv.org/abs/2012.02417
- https://arxiv.org/pdf/1802.09477.pdf
- https://emanual.robotis.com/docs/en/platform/turtlebot3/machine_learning/
- https://github.com/toxuandung/DRL_Navigation_Robot_ROS2_Foxy/tree/main
- https://github.com/sgawalsh/dqnTurtlebot?tab=readme-ov-file

Video Result Links:

- DQN's performance at 300 epochs: Video Link
- DQN's performance at 500 epochs: Video Link
- DQN's performance at 500 epochs with hyperparameter tuning: Video Link
- TD3's performance at 300 epochs: Video Link
- TD3's performance at 500 epochs with hyperparameter tuning: Video Link