ME5418 Project Final Report

Guided Pushing with 7-DOF Robotic Arm using Reinforcement Learning

Group Number: 27

Name: Lam Chun Pok

Metric Number: A0305085J

Contents

1. Introduction

1.1 Background

In recent years, autonomous task handling has become one of the most heated and focused topics, with major breakthroughs and advancements in both hardware and software, including the popularization of robotics arm commercially, and autonomous workstations brought up in the industrial 4.0 movement. Without a doubt, we are developing towards a world where more and more repetitive and dangerous tasks are able to be programmed and designed for robots to perform. With more and more companies investing and implementing autonomous tasks with minimum human-intervention, robots are being asked to perform more tasks at higher capabilities and complexities to step up and cover human work. Robotic arm is one of the most implemented robots, robotic arms are usually made up of multiple links, connected by joints which each provide a Degree of Freedom (DOF). For robotic arm to perform complete representation of motion in the cartesian plane (X,Y,Z,RX,RY,RZ) the robotic arm has to possess at least 6 DOF/joints. However, with only 6 DOF, the workspace of the robotic will be limited with multiple singularity points, therefore robotic arms with more DOF are being implemented to provide a better workspace with less singularity points. However, as the number of DOF increases, the complexity of the control will increase in an exponential fashion, making the compute and design of the control for the robotic arm to be in great scale and a huge load. Furthermore, to exploit on the capability of high order complex robotics, basic human actions such as grabbing, pushing and placing are being mimicked to start with. For this project, the use of a 7-DOF robotic arm to perform the action of guided pushing of a ball to a hole is being investigated.

1.2 Existing Methods Evaluation

More simpler ways of robotic arm control method include algorithms like Rapidly exploring Random Trees (RRT) or Probabilistic Roadmaps (PRM). These algorithms can plan robot trajectory tasks, however these algorithms may struggle in dynamic environments, where the goal position and object types might change. Model Predictive Control (MPC) is also commonly used to predict future states, with excellent precision and optimization upon a well designed model, however for high-dimensional space such as a 7-DOF robotic arm in our task, it will become an extremely computationally expensive approach. For our task, reinforcement learning (RL) is a more suitable and reliable method as it enables the robot to learn optimal behaviors through interactions and rewards.

1.3 Scope

In this report, we will be introducing our proposed reinforcement learning method in completing the task of guided pushing of a 7-DOF arm. Detailed framework, learning agent and neural network structure will be covered, and providing with our results and discussion evaluation.

2. Proposed Method

2.1 Covered Sections

The full RL Cast, learning agents and training procedures have all been covered in our previous entries of OpenAI Gym Code Report and Learning Agent C&R. No new changes have been made for the above sections after the submission.

2.2 Neural Network Structure

The neural network structure for our proposed solution has been updated since last submission of the Neural network C&R. A multi-headed attention layer is added after the observation processing and the goal processing networks. This can help improve the network's ability in focusing on relevant observation-goal relationships dynamically.

Also, residual blocks for residual connection are added after the actor and critic networks. This supports the development of deeper architectures by addressing the vanishing gradient problem and promoting the reuse of features learned in earlier layers. Our residual block includes a fully connected layer with an input and output size of 256, along with LayerNorm and ReLU activation.

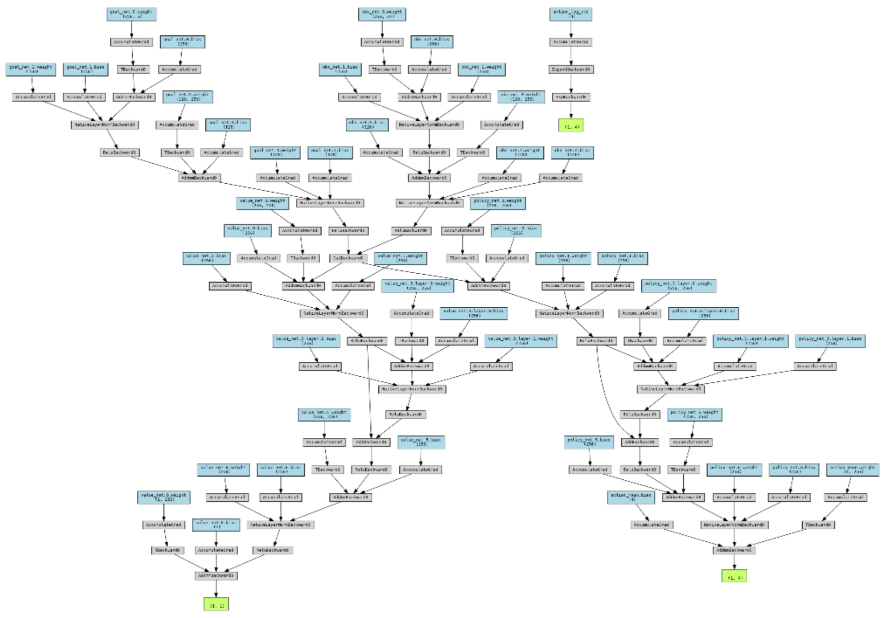


Figure 2-1 Neural Network Architecture

3. Training Evaluation

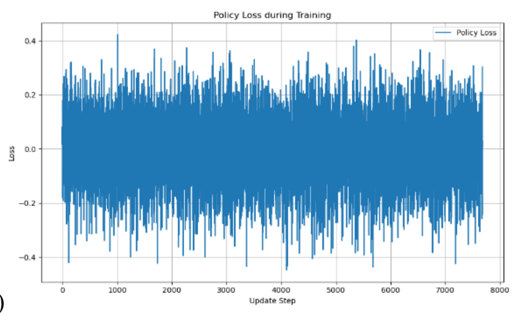
3.1 Training Approach

The proposed solution is being trained on 500 episodes and 100 steps on different reward structures. Training is done and started upon running the setup_ppo.py

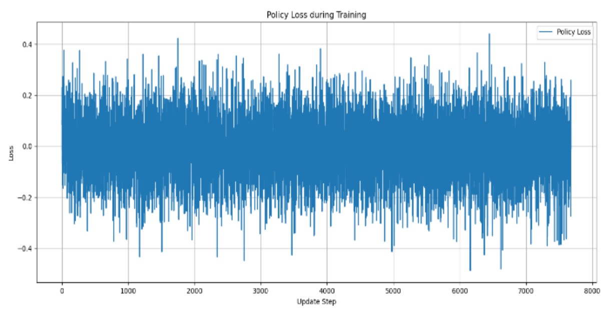
Details are as follows.

3.1.1 Policy Loss

The training data is displayed as below



(a)



(b)

Figure 3-1 Policy Loss Graph (a) Sparse reward (b) Dense Reward

Upon observing and comparing the policy loss graph, we can see that they have similar performance on the policy loss. While giving them a closer look, it is observed that the policy loss for sparse has a larger variance compared to that of the other. Also, the policy loss for the dense reward shows a better structure, showing a better stability

3.1.2 Total Reward

At first training is performed with the reward structure set at dense, and the position of the ball to spawn at random positions on the surface.

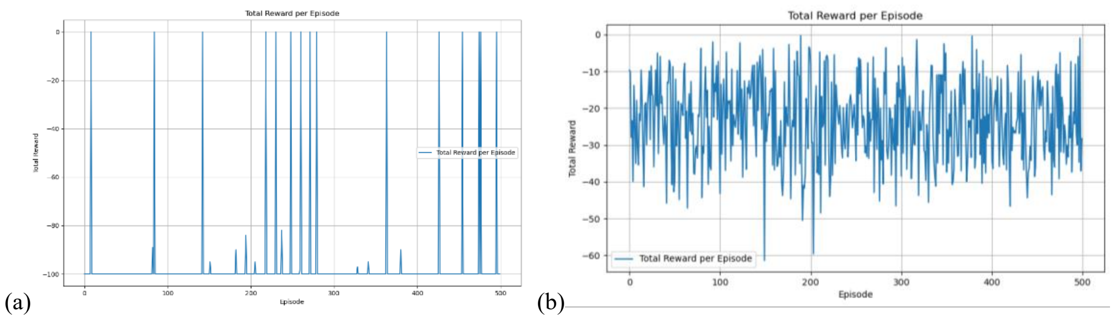


Figure 3-2 Total Reward per Episode Graph (a) Sparse Reward (b) Dense Reward

Unlike the policy loss graph, the total reward graph shows big variations. On sparse reward, the total reward is constantly at the minimum with only a few instances reaching a high reward, showing its ineffectiveness in learning, where the performance is similar to an untrained policy. And for the dense reward, it is observed that the reward for each episode improves from the baseline, and remaining at the certain range, this shows that the policy has been updated effectively to perform better, however the reward has no obvious trend of converging to zero, showing that training is still insufficient, and the policy is still unreliable.

3.1.3 Total Loss

Reward structure is then changed to sparse, and retrained

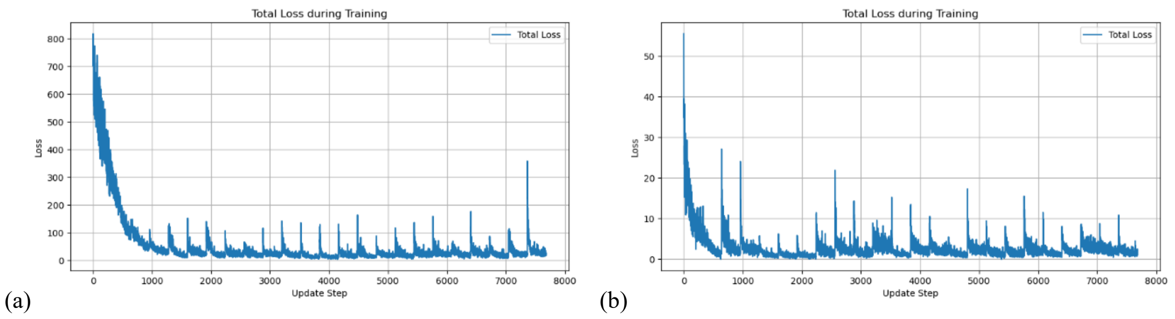
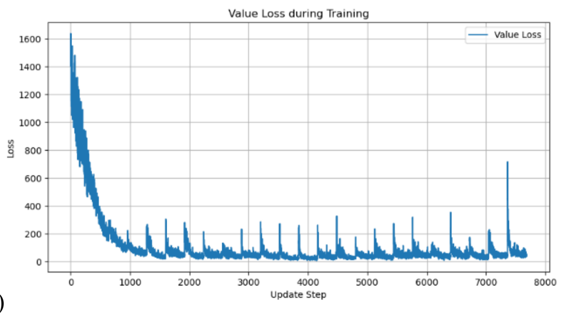


Figure 3-3 Total Loss Graph (a) Sparse Reward (b) Dense Reward

Upon comparison, we can see that the three graphs have similar structures, where total loss has high initial value and rapidly decreases. However we can see that, the total loss for sparse reward requires a longer step time to reach to close to zero while dense rewards only require a lesser step time to close to zero, showing that dense reward has a better learning efficiency than that of sparse rewards. Furthermore, the sparse reward also has a higher total loss value throughout the policy and a more rapid oscillation behaviour over dense reward, showing that dense reward has a better performance and stability over sparse reward.

3.1.4 Value Loss

(a)


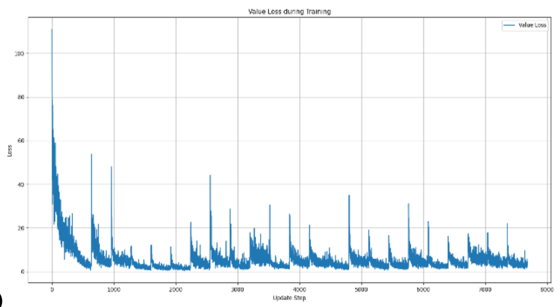
(b)


Figure 3-4 Value Loss Graph (a) Sparse Reward (b) Dense Reward

Value loss graph also possess the same properties and characteristic as to the total loss graph, where sparse rewards have a slower response, higher loss value and oscillation when compared to dense rewards

3.2 Summary

To conclude the training process, it is observed that sparse reward structure is not suitable for our project, where dense rewards structure provides a much better performance in terms of learning efficiency, stability and performance in completing the task. At the same time, it also shows that our designed model is effectively learning and updating its policy to achieve a stable and reliable policy. However, with the complexity of the task with random initial position of both ball and hole, 500 episodes is obviously too little to come up with a usable model, and requires way more training which may take up countless hours. Therefore we simplified the environment to get a glimpse of the performance of our designed model. We performed training with the same design, only changing the position of the ball and the hole from random to a fixed position, and have it trained for 1000 episodes.

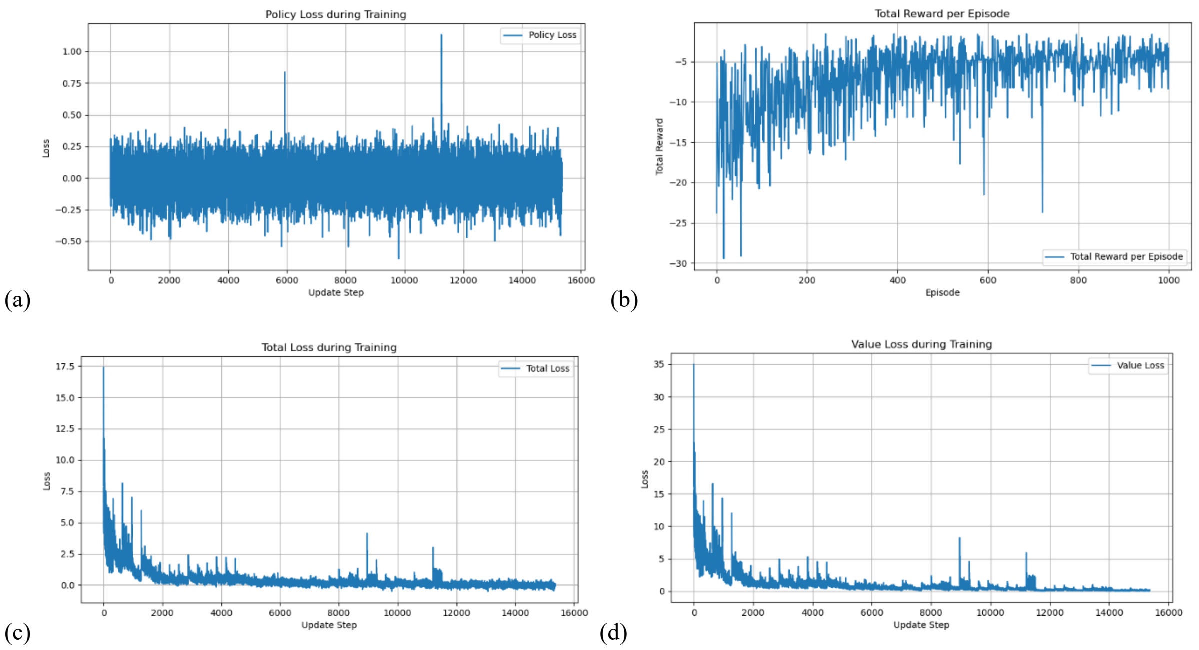


Figure 3-5 Simplified Model Training (a) Policy Loss Graph (b) Total Reward per Episode Graph (c) Total Loss Graph (d)Value Loss Graph

It is observed that when compared to random positions, fixed positions provide a better and more obvious learning performance, with a clear trend of converging towards 0 in the total rewards, this shows that the policy is effectively learning and updating, with high performance of the policy. Also, for both total loss and value loss, the response time, oscillation range and loss value all perform better than that of random point, showing its learning efficiency, performance and stability to be better.

4. Results and Discussion

4.1 Results

Our model of dense reward structure with fixed initial ball position is set to be evaluated on 100 episodes. The result is as follows.

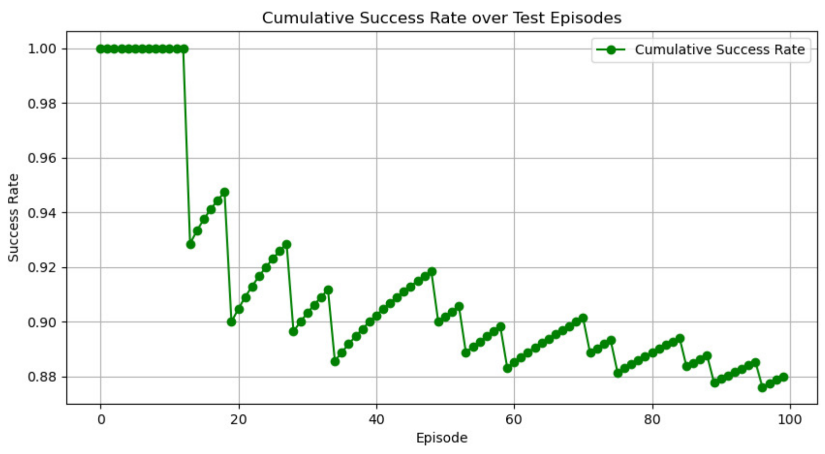


Figure 4-1 Cumulative Success Rate over 100 Episodes

As shown in the graph, our model is able to reach a success rate of 88% over 100 episodes, where the ball is successfully guided and pushed into a hole. This proves our evaluation and conclusion made on the training process on its reliability and stability.

4.2 Comparison with conventional baselines

The same environment is being evaluated in a stablebaseline3 PPO in **evaluate_model_sb3.py**, the result from stablebaseline3 PPO shows similar result in success rate in completing the task. This shows that the proposed solution and our model was accurately designed performance-wise. However, upon evaluating the performance in the simulation visualization, it is observed that our model moves in a fast but shaky manner when compared to stablebaseline3 PPO, which may possibly cause flaws in implementation.

4.3 Reflections

As a mechanical engineer student with little background of coding, attempting this course is a big step out of my comfort zone. Before the course I had no idea on how machine learning is done and more importantly how it can be implemented onto robotics. Throughout the course I have grown to learn about the basics of machine learning and how they are applied to robotics. This project has been a struggling one for me, but I still get to try out how machine learning can be integrated into robotics. Being my first ML project, I have learnt a lot throughout the project, from the basic of choosing architectures to designing the neural network structure, and most satisfyingly seeing our model truly performing the task as designed. Thanks must be given to Prof. Guillaume and the TAs for the support and teaching, and also my project teammates that supports me throughout the project and letting this project works.