ME5418 Machine Learning in Robotics

Autonomous Fruit Picking Robot

Group 7 Final Report

Student name : Lin Haoyu

Student Number : A0304971A

Teammates : GAO Yilin, SUN Yining

22nd November 2024

1 Introduction

This project focuses on autonomous fruit picking robot implementation of algorithm.
In the last task, we completed the implementation of ppo2agnet. On this basis, we rediscussed and sorted out the tasks that we needed to improve in the final stage.
1) Modify and debug the RL framework, adjust the map to a suitable size, and modify the model urdf file accordingly.
2) Modify the neural network to simplify the redundant structure.
3) Train the model, improve the hyperparameters according to the training results, and iterate the optimization.
4) Comparison with traditional algorithms (A*)
5) Render and record simulation video.

2 Modify RL framework

The first phase of this part involved modifying the Gym environment to ensure compatibility with our specific the need for models to converge quickly. The map size was adjusted to a suitable dimension, balancing computational efficiency and task complexity. We previously set the world map to a 50*50*15 3D grid world, but the large environment made it difficult for the model to converge. So we changed the world map to a 10*10*10 size. Moreover, in our_gym_easy, we temporarily removed the EXTRACT action from the action space, so that we could observe the training effect in a simpler environment and find the direction for improvement, and then re-add the EXTRACT action in the full version of our_gym.
Corresponding changes were made to the model's URDF file to align with the updated map specifications. These adjustments included changing the robotic arm adjustable length, fruit size, trunk size, which improves the visual effect when rendering.

3 Modify neural network and learning agent

We reexamined the neural network. and remove redundant structures that could decrease training efficiency and performance. We modify the structural parameters of each layer of the neural network to adapt to different types of inputs. Layers unnecessarily complex were simplified or removed without compromising the model's learning capacity.
In our initial considerations, we identified a potential limitation in the PPO2 algorithm: its lack of an inherent memory module. To address this, we proposed integrating an LSTM network to serve as a memory module, enabling the model to better handle temporal dependencies in sequential decision-making tasks. However, during the implementation process, we gradually discovered that the Old Actor Network, a core component of PPO2[1], inherently fulfills the role of a memory module.It retains historical policy information, effectively allowing the model to account for past experiences and improving its decision-making capabilities over time. Given this

functionality, the addition of an LSTM network became redundant. Its presence not only added unnecessary complexity to the architecture but also increased computational overhead without providing significant performance benefits. As a result, we decided to remove the LSTM module from our implementation, simplifying the network while maintaining its effectiveness.

This adjustment shows that we are gaining a deeper understanding of the intrinsic mechanisms of reinforcement learning algorithms. This step streamlined the architecture, reduced computational overhead, and improved training stability. The resulting model is more interpretable and computationally efficient, enabling faster iterations during the training phase.

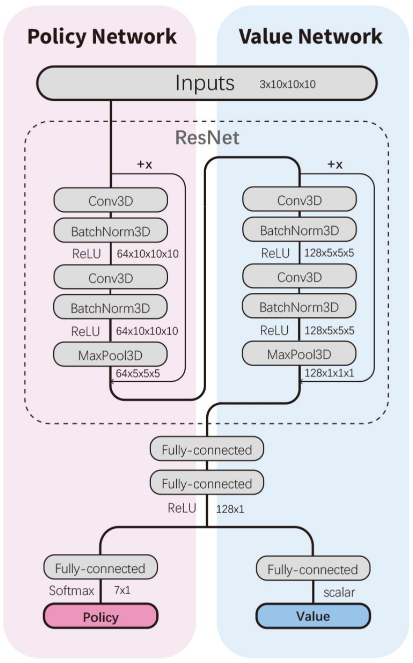

Figure1 NN structure

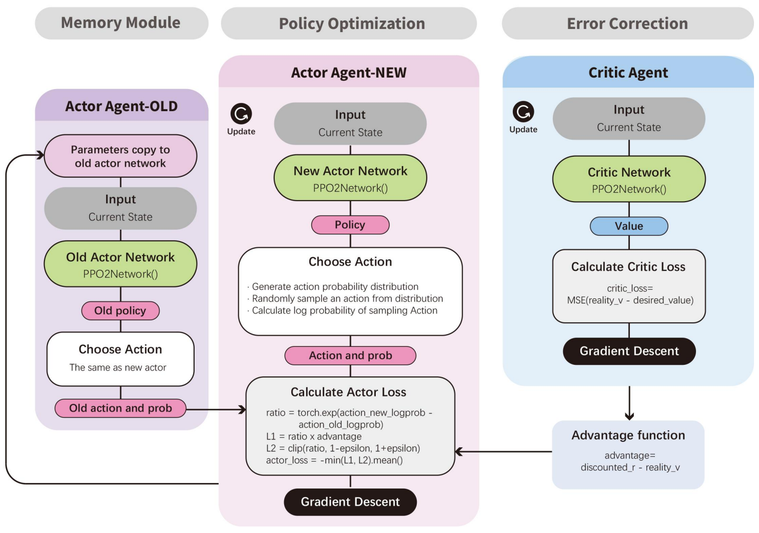

Figure2 learning agent structure

4 Training and optimization

Table1 Hyperparameters

Hyperparameters	Policy1_easy	Policy2	Policy3
EP_MAX	1000	1000	1000
EP LEN	2000	2000	2000
GAMMA	0.8	0.8	0.6
A_LR	0.0001	0.0005	0.0001
C_LR	0.0003	0.0008	0.0003
A UPDATE_STEPS	10	10	10
C_UPDATE_STEPS	10	10	10

We use the modified Gym environment and the updated neural network architecture for training. During this phase, the model's performance was evaluated based on training results, such as loss convergence, reward accumulation, and episode length. Hyperparameters, including learning rates, discount factors, batch sizes, and update steps, were iteratively tuned to enhance the model's learning process.
By observing the results of simplified gym, we find that policy loss fluctuates greatly and is difficult to converge, while value loss converges slowly. Considering the actual situation, we believe that the fruit-picking robot should consider picking the expected amount of fruit as soon as possible in each working cycle, so it should pay more attention to the most recent fruit, that is, the agent should pay more attention to short-term gains, so the discount factor GAMMA should be smaller. Since Critic is usually more sensitive to the learning rate, we lowered C_LR to 0.0002, and Actor might need a slightly higher learning rate to update the policy faster, we adjusted A_LR to 0.00015.We will reduce A_UPDATE_STEPS to 5 to avoid overoptimizing the current policy with the actor. We raised C_UPDATE_STEPS to 15 to make sure the critic fits the target value better.
By experimenting with greater learning rates, we stabilized the training process and reduced oscillations in the loss functions. Adjusting the discount factor allowed the agent to better balance short-term and long-term rewards, resulting in more efficient policies. Following these adjustments, the model demonstrated significantly better training performance. The reward curves exhibited consistent improvement across episodes, with reduced variance. Moreover, the agent's actions became more deliberate and aligned with the desired objectives, indicating a stronger ability to learn optimal policies.

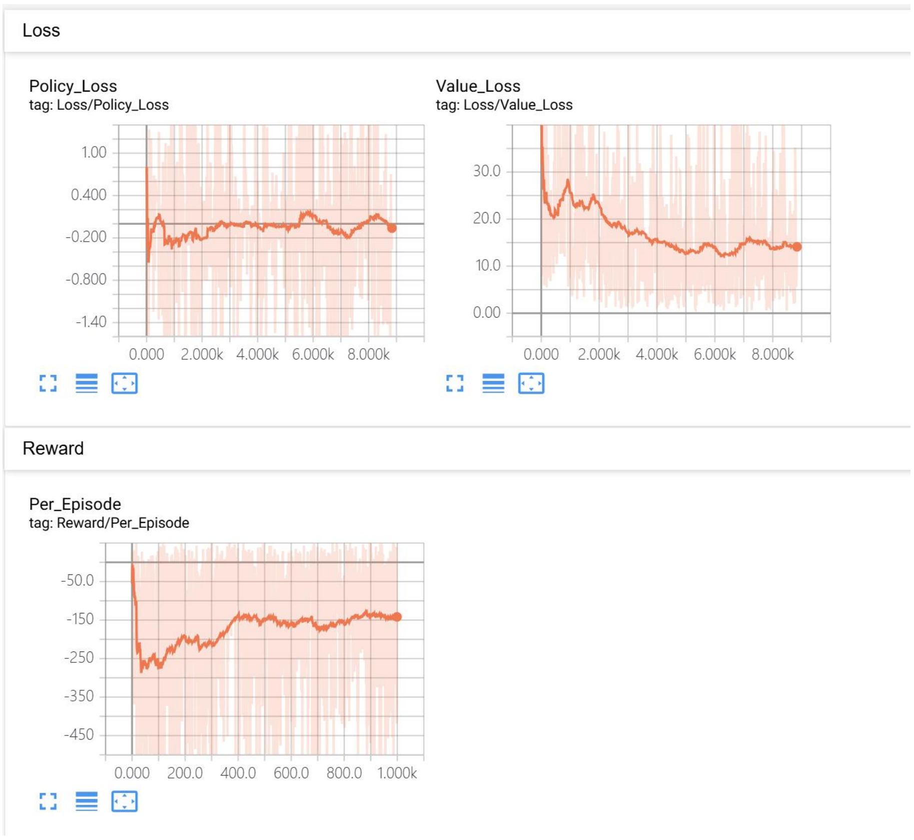

Figure3 Policy1_easy

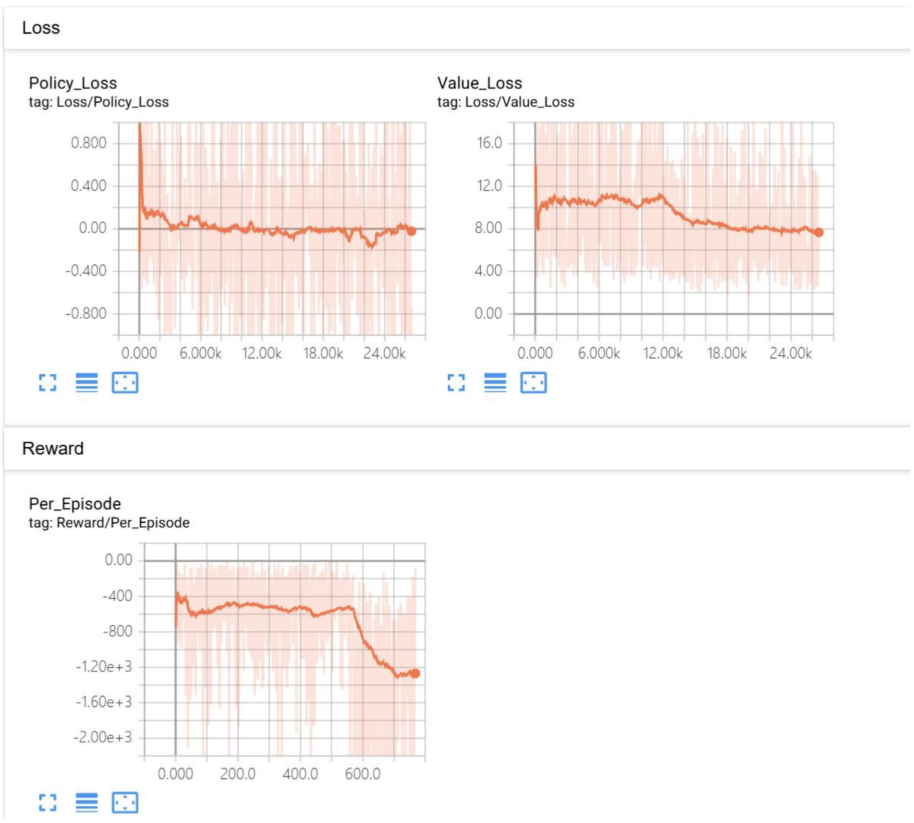

Figure4 Policy2

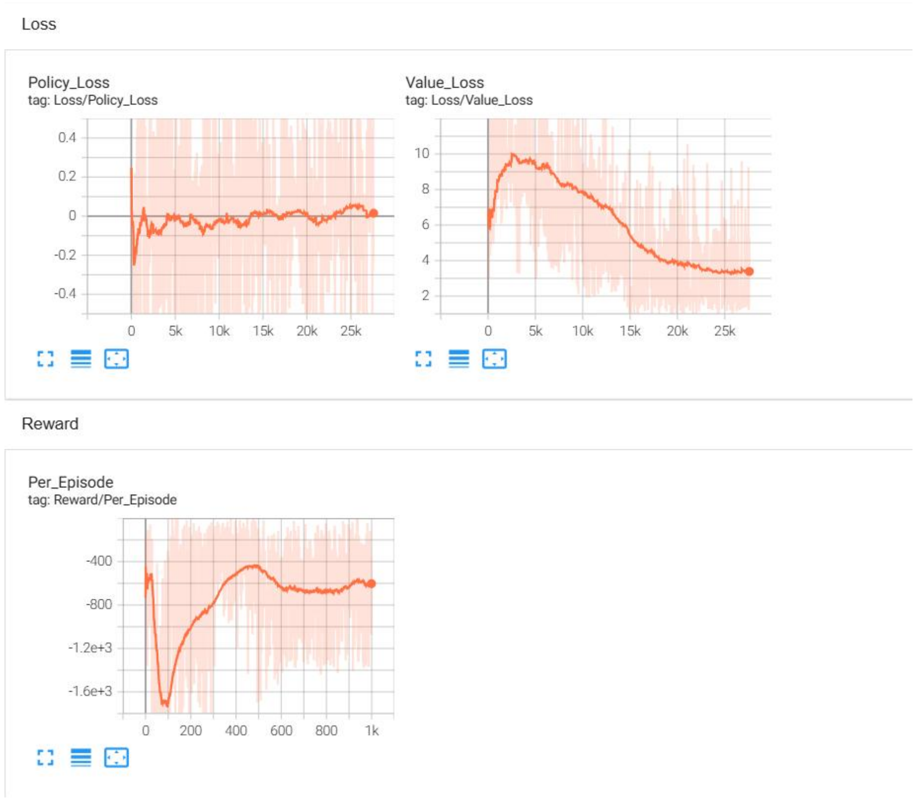

Figure5 Policy3

5 Discussion and conclusion

1) Advantages and limitations

The 3D grid-based environment supports random generation of obstacles, goals, and agent initialization, making it adaptable to various scenarios. And we set well-defined reward mechanism that encourages the agent to perform optimally while penalizing undesirable actions like collisions or exceeding limits. Features like height limitations, tree canopy fruit generation, and collision detection contribute to realistic simulations. The implementation demonstrates adaptability to the FruitPickingEnv, leveraging neural networks for learning state representations.

The implementation can become computationally expensive for larger grids or more agents due to nested loops and multiple assertions. Training involves multiple iterations of backpropagation, leading to significant resource consumption.

2) Challenges encountered

We also encountered many difficulties during the project. The biggest difficulty we encountered was that because of the complexity of the action space, it was difficult for the robot to distinguish which action was efficient, resulting in difficult convergence of the model. We tried many ways to solve this problem, we first reduced the world map and simplified the action space, experimented in simple mode, explored the direction of improvement, and then re-complicated the action space for further experiments.

Another difficulty we encountered was that the initial training results were not satisfactory. In this regard, our team members used different hyperparameters for training, and summarized and compared, so as to find the direction of optimizing parameters.

3) Comparison with traditional algorithms (A*)

We use a reward-based system, which may require more exploration compared to A*'s heuristic driven approach[2]. A* is more efficient for finding the shortest path to a static goal, while our implementation excels in complex, multi-goal scenarios.
a* algorithm is a traditional algorithm for shortest path planning. Compared with RL algorithm, a*s algorithm requires agent to have global vision. The a* algorithm is generally used to deal with two-dimensional space, but it is not good at dealing with three-dimensional space. For the 3D discrete space in this project, the problem can be separated into 2D+1D, first move in the plane, find the nearest fruit projection point, and then vertically expand the arm to change the height and absorb the fruit after reaching the position. The concrete method is to first project the 3D coordinates of the fruit to the floor plane, and then make the center coordinate of the base and the projection coordinate difference, calculate the action list, and move the base to coincide with the projection coordinate. Then get the difference between the z coordinate of the fruit and the z coordinate of the end effector, get the action list of the robot arm, and expand the robot arm to the specified position to pick the fruit.
Because a* can only have a better planning effect in the known and determined environment in the calculation, the environment needs to be simplified in order to use the a* algorithm. In order to prevent dynamic changes in the map, we remove the ripen of the fruit and do not detect whether the fruit has been picked. Moreover, a* can only pick fruit for a single time, and it cannot pick multiple fruits continuously in a single process. PPO2 inherently balances exploration and exploitation, whereas A* is a deterministic search algorithm that finds an optimal path based on a predefined heuristic. Therefore, in the process of picking fruits in this project, once the number of fruits is larger, RL method will have more potential to deal with more generalized scenarios.

4) Lessons learned

Through this project, we gained a deeper understanding of reinforcement learning environments. We learned to create complex 3D environments, manage agent positions, and represent goals and obstacles effectively, ensuring consistency and accuracy in state transitions. Setting a reward system taught us the importance of balancing penalties and incentives to guide agent behavior towards desired outcomes. Defining a flexible but constrained action space for agents reinforced the need for logical mappings between actions and their effects in the environment.
We successfully implemented PPO2, experimenting with key elements like the actor-critic architecture, policy clipping, and reward discounting. This gave us practical experience with advanced optimization techniques and their difficult challenges. By applying PPO2 to a Fruit Picking Environment, we explored how reinforcement learning can solve practical tasks, understanding the importance of environment design in determining model performance. We overcame technical issues such as

unstable gradients and network convergence, enhancing our debugging and optimization abilities. Through TensorBoard We visualized the Loss function image to display the training results more intuitively. This project enhanced our theoretical and practical knowledge of reinforcement learning while fostering enlightening thinking and problem-solving skills. At the same time, this project has also improved our teamwork ability, and what is more valuable is that we have gained a profound friendship.

5) Future works

We will optimize the structure of the gym so that it can handle more complex environments and allow the robot to have more abundant movement space. We will also improve the structure of the neural network, so that it can analyze the value of policy more accurately, and optimize the logic of agent's choice action to make it more efficient. In addition, we also considered optimizing the robot's model and adopting more realistic simulation methods to make the rendered visualizations look more vivid.

Reference

[1] PPO2 algorithm: https://zhuanlan.zhihu.com/p/538486008
[2] Function A*: https://github.com/while-TuRe/A-star-ShortestPath