Hindsight Experience Replay with Environment Relabeling

Milestone Report

Katharina Hermann
Technische Universität München
Department of Informatics

Ferenc Török
Technische Universität München
Department of Informatics

***Abstract*—This document is written as a milestone report for the project of the subject Advanced Deep Learning in Robotics at the Technische Universität München. Our chosen topic is to develop a method to train a Neural Motin Planner (NMP) [2] agent with Reinforcement Learning (RL) [1] in an environment with obstacles. The aim of the method is to reduce training time as much as possible meanwhile using a very simple reward function. For this we use Hindsight Experience Replay (HER) [3], [4] with environment relabeling.**

I. Experiment Setup

Agent: For now we have used a point-mass robot in 2D for the sake of simplicity. The robot has got a radius. If an obstacle is closer than the radius, a collision occurred. Also, if the goal is within the agent's radius, it is considered to be reached. The agent has 2 continuous actions with which it is able to interact with the environment, a x and y direction step distance. These can be arbitrary values between their respective maximum and minimum values.

Reward specification: The task of the agent is to navigate from one point to an other in a workspace which contains some obstacles which should be avoided. Or reward function is as simple as it gets: the agent receives a reward in the amount of $+5$ if the it has reached the goal, -1 if it has collided with an obstacle or left the workspace and -0.01 after every timestep.

Workspace: Workspaces are represented as a 32×32 grid, where each grid cell is either free space (value 0) or obstacle (value1). The number of obstacles, their position and size are generated randomly.

Start, goal: Start and goal positions are also generated randomly for a given workspace. A workspace with the obstacles, the goal and the agent is illustrated in Figure 1.

Performance metric: We use success rate as the evaluation metric. In every evaluation cycle, we carry out N evaluation episodes. The succes rate for every evaluation cycle then calculated as follows: $\mu = N_{succ}/N$, where N_{succ} is the number of successful episodes.

II. Implementation

We have chosen the agent to be trained with Deep Deterministic Policy Gradient (DDPG) [5] which is an Actor-Critic algorithm for continuous state and action spaces. We use an off-the-shelf implementation, the python package TF2RL [6].

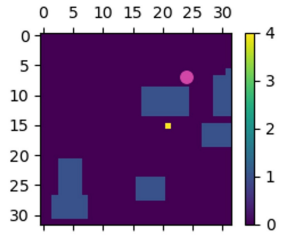


Fig. 1. Workspace with obstacles (light blue), agent (pink) and goal (yellow)

This package implements various RL methods in Tensorflow 2.x. For the environment we have implemented a custom Open-AI gym environment.

The main building blocks of the project are as follows:

- Workspace and goal generation.
- Convolutional Autoencoder (CAE) for reducing the workspace representation size. This is necessary since the workspace in its original form is represented with a feature vector of size 1024. Compared to this, the state of the robot and the goal are represented with $2-2$ features. The workspace feature vector reduction is necessary not to loose the information about the state and the goal position.
- Environment implementation as a custom Open-AI gym environment.
- Workspace relabeling implementation
- Implementation of the trainer class with HER and environment relabeling.
- Tests

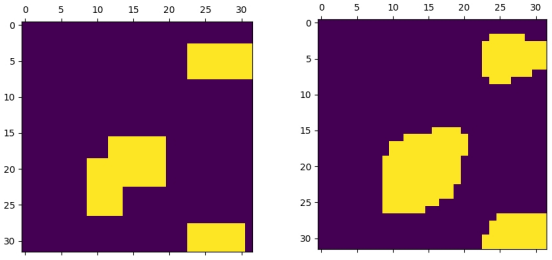
III. Progress

So far we have made the following progress:

- Implemented and tested workspace, start and goal generation.
- Implemented, tested and trained the CAE on 10000 workspaces. For the encoder we have used 3 Convolutional layers with filter sizes of $[4, 8, 16]$ followed by max-pooling for size reduction. Then after a linear layer the feature size was reduced to the latent space dimension, which is 16. For the decoding we have used a linear layer followed by transpose convolutional and

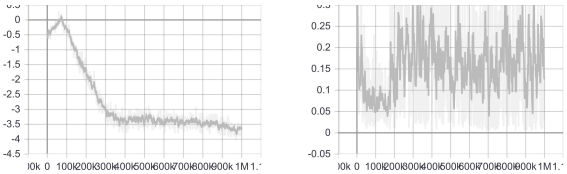
convolutional layers. We have used weighted binary cross entropy loss for training. With this method we have achieved 96% accuracy on the test set. An example of the original and the output image can be seen on Figure 2.

- Implemented and tested the custome Open-AI gym environment
- Implemented and tested a simple relabeling method. The method so far is very simple, it only removes the obstacle into which the agent has collided or shifts the workspace and the trajectory if the agent has left the workspace. The new goal is the last state of the trajectory.
- Implemented and tested the training method with HER and workspace relabeling. So far we are looking for good parameters for training. We have trained some agents but the results so far are not sufficient. The actor and critic losses and the training and testing success rates of an experimental training can be seen in Figure 3 and 4 respectively.



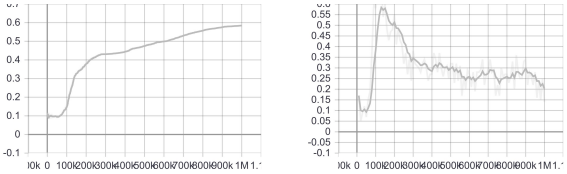
(a) Input (b) Output

Fig. 2. The input and output of the trained Convolutional Autoencoder on a test image.



(a) Input (b) Output

Fig. 3. The input and output of the trained Convolutional Autoencoder on a test image.



(a) Input (b) Output

Fig. 4. The input and output of the trained Convolutional Autoencoder on a test image.

IV. Future work

As it was said previously, the next step is to find the good hyperparameters for training. If we managed to train the agent sufficiently, we will evaluate the results. We have in mind to compare the trained agents performance to a benchmark agent. For the benchmark agent, the reasonable choices would be for example a DDPG agent without HER and the agent that was trained with the method explained in [2].

If there will be enough time remaining, we will experiment with more sophisticated relabeling methods and possibly try to apply the whole method to a more complex robot, for example to a robot arm.

References

[1] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," 2011.

[2] T. Jurgenson and A. Tamar, "Harnessing reinforcement learning for neural motion planning," *arXiv preprint arXiv:1906.00214*, 2019.

[3] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in neural information processing systems*, 2017, pp. 5048–5058.

[4] A. C. Li, L. Pinto, and P. Abbeel, "Generalized hindsight for reinforcement learning," *arXiv preprint arXiv:2002.11708*, 2020.

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[6] K. Ohta, "tf2rl package for python," 2019. [Online]. Available: https://github.com/keiohta/tf2rl