Offworld Assessment Documentation

Siddharth Ghiya

September 2020

1 Setup the Environment and Run the Test

I tested both the simulator environment and real environment. Unfortunately, I didn't record a video of testing on the real environment. The video of the robot taking random actions in the simulator environment can be found by following the link : `https://github.com/siddharthghiya/offworld/tree/master/videos/part_1`.

2 Try to Solve an Offworld Gym Environment End-to-End

2.1 Training

I used Proximal Policy Optimisation(PPO) to solve the task. I chose to use PPO because it is an on-policy algorithm and previously only an off-policy algorithm(DQN) has been evaluated on the environment. Since on-policy algorithms typically require more samples than off-policy algorithms, I have used parallel environments (only in simulation) to fasten up the training process. I looked at the network used by engineers at Offworld and used that as an initialisation for designing the network structure of the agent used in the experiments. I have used greater number of parameters in the initial layer. The number of parameters can be decreased to fasten up the training process but more experiments need to be performed to effectively decide the size of the deep network. All of the code used by me can be found at : `https://github.com/siddharthghiya/offworld`. The folder "ppo" contains the code for PPO. I have used a pytoch based implementation of PPO for this task. The implementation I used can be found at `https://github.com/ASzot/ppo-pytorch`. To train the agent, you can run the "main.py" file in the main directory. Various arguments need to be provided to train the robot. Different arguments are as follows:

1. **num-envs**: I used parallel environments to train the agent. This argument can be used to specify the number of parallel environments. (Default : 8)
2. **real**: A boolean to specify if the training is being done on a real environment. (Default : False)

3. **load-dir**: To continue training from a previously stored checkpoint, load-dir can be used to provide the path of the saved model and weights. This model and corresponding weights are then loaded and used as an initialisation for further training. (Default : None)

4. **exp-name**: Name of the experiment. Also used as sub-directory name within the "weights" and "runs" directories to keep track of the experiment. (Default : tmp)

5. **channel-type**: The type of observation to the agent being trained. You can specify the input channels as RGB_ONLY or DEPTH_ONLY. (Default : DEPTH_ONLY)

6. **save-interval**: Frequency of updates at which the model weights are stored. (Default : 10)

7. **log-interval**: Frequency of updates at which the training performance is logged. (Default : 1)

8. **num-steps**: Number of steps in between every update of the agent. (Default : 100)

   For this task, I have trained the agent only the "depth channel". To reproduce the results please run the following command from terminal:

   python main.py −−exp-name depth

   The different curves are as follows:

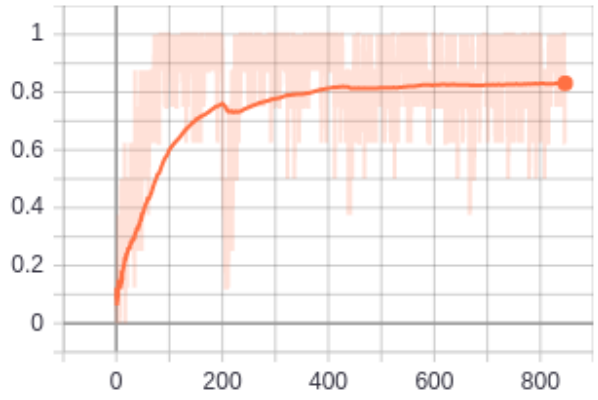

Figure 1: Reward curve obtained during training

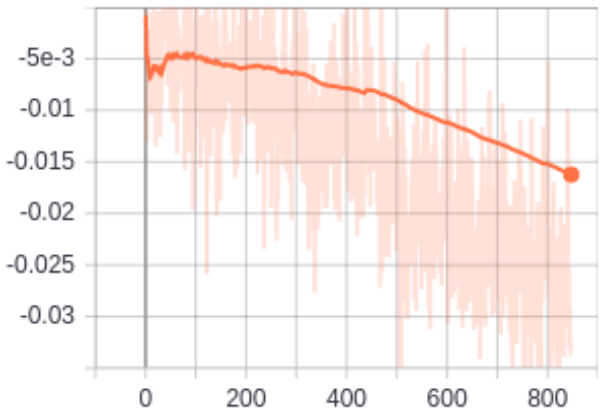

Figure 2: Action loss vs updates

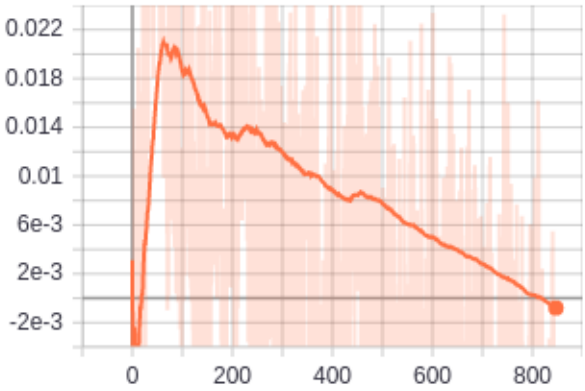

Figure 3: Value loss vs updates

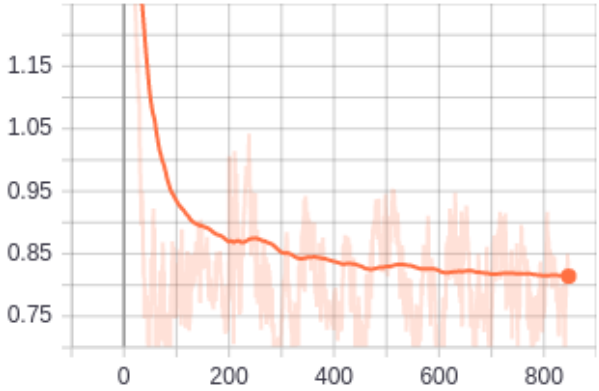

Figure 4: Entropy loss vs updates

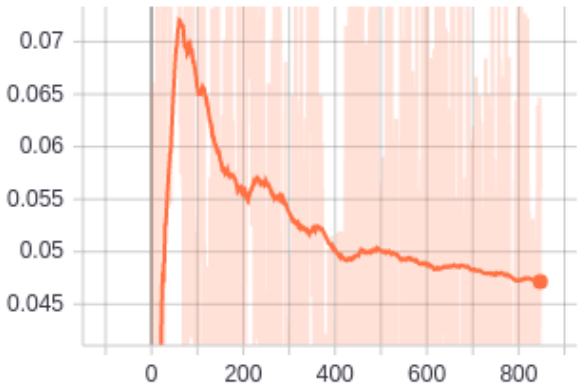

Figure 5: Overall loss vs updates

2.2 Evaluation

I evaluated the trained agent and have recorded a single episode of the trained agent. You can find the video at `https://github.com/siddharthghiya/offworld/tree/master/videos/part_2`.

I further recorded the success percentage and the average number of steps per episode over 30 episodes. The average number of steps per episode were 27.46 and the success percentage was 96.67.

You can evaluate the trained policy by running the following code from terminal:

python evaluate.py −−load-dir weights/depth/model_390.pt −−num-eval 30

3 Train an Agent on a Physical Robot using Sim-to-Real Transfer from Your Previous Agent or With Human Demonstrations

3.1 Training

I tried to train the agent in a real environment by using the policy trained in simulated environment as an initialisation. However, the server went down and did not come up till the end of the challenge. I was only able to perform 18 updates to the policy and was not able to train the agent completely. Given more time, I think I should be able to train the agent using PPO in the real environment.

To train on the real environment using policy trained on the simulator as an initialisation, please run the following command from the terminal:

python main.py −−exp-name real_depth −−load-dir weights/depth/model_390.pt −−num-envs 1 −−real −−log-interval 1 −−save-interval 2 −−num-steps 35

If the connection to the server breaks down while training, just provide to the path to the latest trained weights in the "load-dir" argument.