Tackling Obstacle Tower Challenge with Random Network Distillation

Hanchul Choi, Stanford University CS230 Course Project

Video Link: https://youtu.be/CLGVLRLFAB0

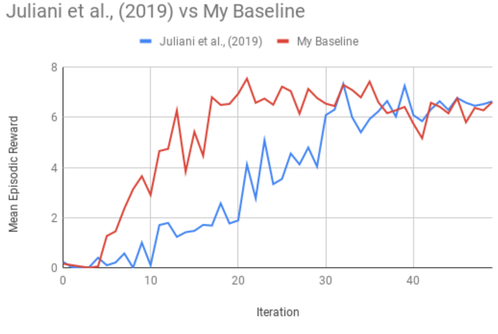
The Problem

The Obstacle Tower is a reinforcement learning environment created by Unity, where an agent has to climb a tower that becomes increasingly difficult as the level advances. The environment provides a comprehensive benchmark that tests computer vision, locomotion skills, high level planning, and generalization. (Juliani et al., 2019)

The goal of this project is to train a reinforcement learning agent using the materials learned from CS230 course.

Environment and Data

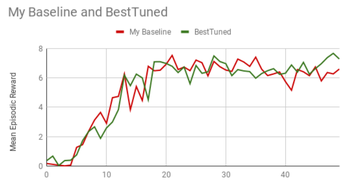
The state space is a 168x168 pixel RGB image with four stacked frames to capture the temporal element. The action space consists of 54 combinations of moving, jumping, and camera rotation. Reward is given when the agent opens a door or goes up the floor. I preprocessed the data into greyscale 84x84 pixel image and removed unnecessary actions from the action space.

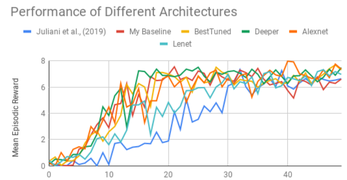


The baseline model for the project is a Deep Q-Network with several improvements including prioritized replay, distributional network, and multistep update. Preprocessing improved the convergence as above.

Approach

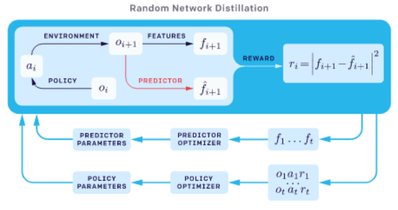
First, I attempted to tune the hyperparameters of the baseline Deep Q-Network model, including the learning rate, type of optimizer, discount gamma, and epsilon decay rate. The best tuned model (left figure below) only showed marginal improvement.





Next, I implemented different neural network architectures learned in class such as Lenet and Alexnet to improve the computer vision aspect of the agent. (right figure above) Unfortunately, this approach also showed only marginal improvement. After a several rounds of human play, I found that the bottleneck was locked doors that required keys, meaning that high level planning was more important than computer vision.

The third approach I took, therefore, was to use sequential models that could account for the history of agent's actions. This enabled the agent to learn the relationship between the key and the locked door even though the two events were multiple frames apart. This greatly improved the agent's performance.



source: OpenAI

In addition to these standard improvements to the baseline model, I implemented a Random Network Distillation (Burda et al., 2018) This algorithm introduces synthetic reward for exploring novel states that the agent did not experience, thus solving the problem of sparse rewards. Although the computation is taking more than ten days to run, interim results already went past the previous algorithms' performance.

Performance

Algorithm	Architecture	LR	Action	Performance
Default Paper Rainbow	Nature	0.0000625	54	6.64
Baseline DQN	Nature	0.0000625	12	6.61
Best Tuned DQN	Nature	0.0001	10	7.31
Deeper DQN with one more conv layer	Nature	0.0001	12	7.48
Lenet-like DQN with avg pooling	Lenet	0.0001	12	6.97
Alexnet-like DQN with max pooling	Alexnet	0.0001	12	7.37
VGG-like like	VGG	0.0001	12	N/A
Resnet-like like with residual blocks	Resnet	0.0001	12	N/A
Inception-like like with inception blocks	Inception	0.0001	12	N/A
Inception-resnet-like like	Inc-Resnet	0.0001	12	N/A
RND DQN, raw IR and states	Nature	0.0001	12	0.52
RND DQN, normalized IR	Nature	0.0001	12	3.14(interim)
RND DQN, normalized IR and states	Nature	0.0001	12	8.31(interim)
Baseline PPO	CNN	0.00025	12	6.78
PPO with reduced action space	CNN	0.00025	8	6.72
PPO with tuned with stacked frame	CNN	0.00025	10	6.98
PPO with LSTM	LSTM	0.00025	10	1.41
PPO with CNN and LSTM	CNN/LSTM	0.00025	10	8.27
PPO with CNN and LN-LSTM	CNN/LNLSTM	0.00025	10	7.01

Above is the summary of different algorithms and settings I tried for the project. Since the simplest baseline model takes about 4 days to train on Nvidia Tesla K80, some of the more complex algorithms are still running at the moment, especially the DQN with Random Network Distillation that has to run three separate neural networks.

Next Steps

The Obstacle Tower Challenge sponsored by Unity and Google Cloud is running until July 15th. As I was lucky enough to be selected as one of the finalists, I will be continue my efforts to improve my agent further. The next algorithm I will incorporate is the Go-Explore, (Ecoffet et al., 2019) which was published recently to beat the performance of the Random Network Distillation.

Note: reference has been omitted in the poster due to lack of space, please refer to final report for the list of references.