Fighting Zombies in Minecraft with Deep Reinforcement Learning

Hiroto Udagawa, Ricky Lee, Tarun Narasimhan

Introduction

In this project, we train an agent in the game Minecraft to kill zombies and survive as long as possible using deep reinforcement learning. Our objective is to have the agent learn what policies to execute based solely on the visual information it receives from the raw pixels of the game play screen and the specified reward structure. Rather than giving the agent information about the environment or any guidance on killing zombies, it has to learn optimal actions based on its past experience.

Building on DeepMind's work with deep reinforcement learning, our approach is to use a convolutional neural network and Q-learning to train the agent's behavior over time.

States

- Input: 640x480 pixel RGB images of game screen

- Features: Take last 3 frames and resize and greyscale them into 84x84x3 stack

Actions

- Move Up
- Move Back
- Turn Left
- Turn Right
- No Action

Rewards

- Hit zombie : +5
- Kill zombie: +40
- Lose Health: -5
- Frame Alive: +0.03

{hiroto, sylee1, tnarasim}@stanford.edu

Model

Bellman Equation:

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') \middle| s,a\right]$$

Loss function for Q-Network:

$$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[(y_i - Q(s,a;\theta_i))^2\right]$$

Instead of directly solivng the Bellman equation, we use a convolutional neural network as a nonlinear function approximator to estimate the Q-values. In each iteration, we calculate target y and update our network to minimize the loss function.

Training Process

Initialize game, network, and replay memory

Repeat

Choose next action

With some probability, choose a random action, otherwise choose action with highest Q value from network

Capture image and rewards

Track in-game observations (kills, health, hits) to get reward associated with state transition

Process image into features

Update replay memory

Update the replay memory with new data <s, a, r, s'>

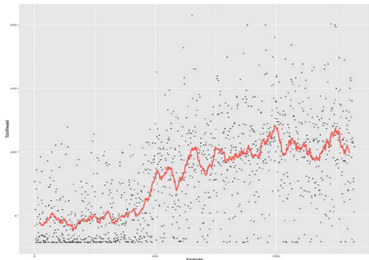
Train network

Sample batch from replay memory and train DQN

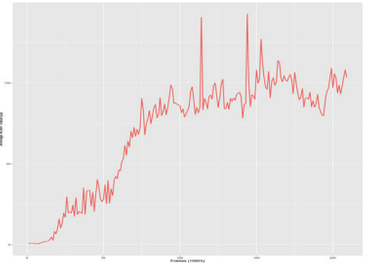
Until Terminated

Results

Cumulative Reward Per Life



Average Q-Value



- The agent showed noticeable improvement in reward per life, beginning with an average reward of around zero for the first 500 lives and increasing to close to 200 by the end of training.

- Behaviorally, the agent started by moving and spinning arbitrarily but ultimately learned to recognize zombies in its field of view and face them to connect hits.

- The right plot shows that over time our agent gets better at picking an action that results in a higher Q-value, which provides an estimate of how much discounted reward the agent can obtain by following its policy.

Future Work

- Tuning the action space to better simulate natural movement and increase precision

- Improve training process by prioritizing more useful experiences from the replay memory

- Running our algorithm on a GPU cluster to improve performance and enable us to run for more iterations

References

[1] Mnih, Volodymyr; Kavukcuoglu, Koray; Silver, David; Graves, Alex; Antonoglou, Ioannis; Wierstra, Daan; Riedmiller, Martin. Playing Atari with Deep Reinforcement Learning, December 2013.
[2] Matiisen, Tambet. Demystifying Deep Reinforcement Learning. December 2015. https://www.nervanasys.com/demystifying-deep-reinforcement-learning/
[3] Chen, Kevin. Deep Reinforcment Learning for Flappy Bird 2015.
[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Mar- tin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, Demis Hassabis. Human-level control through deep reinforcement learning. February 2015.