Defeated Zombies Through Deep Q-Learning in Minecraft

Hiroto Udagawa, Tarun Narasimhan, Shim-Young Lee

December 16, 2016

1 Introduction

In this project, we train an agent in the video game Minecraft with reinforcement learning to kill as many monsters as possible in a confined space. Our agent is only equipped with a sword to attack the monsters, which can themselves attack and kill the agent. The objective is to have the agent learn what policies to execute based on the information it receives about the surroundings and the raw pixels from the game play screen. Using a convolutional neural network and Q-function reinforcement learning, the agent showed definite improvement in its ability to kill monsters. This project displays the promise of combining deep learning with reinforcement learning to achieve non-trivial tasks.

2 Related Work

There have been several projects in this area which we drew upon for our project. Most notably, DeepMind famously trained a convolutional network with Q-learning to play Atari games using raw pixels as inputs and a value function estimating future rewards as output, which ultimately outplayed human experts [1]. [More description.]

[**Ricky: write-up of DeepMind, Flabby Bird guy**]

3 Minecraft and Project Malmo

Minecraft is a popular sandbox video game that was released in 2011. It allows players to control an agent, which can explore the map, build structures, craft tools, and destroy computer-generate monsters. Project Malmo is an open source platform for Minecraft created by Microsoft that offers an interface for researchers to experiment with different approaches to artificial intelligence.

Malmo is a useful environment for artificial intelligence research because it allows users to test endless situations within a relatively simple framework. It allows researchers to control agents' movements and environments using Python scripts. By combining different states, actions, and spaces, researchers can create different environments and define various tasks for which the agent trains with reinforcement learning.

As a simulator, Malmo is more constrained than many other simulators which researchers have employed. Microsoft designed Malmo to be an interface with Minecraft, but not to provide the agent with any information about Minecraft. Thus, the default in Malmo is for the player-controlled agent to not have any information about its environment or the rules of the game. This feature makes Malmo very convenient for reinforcement learning as this replicates the exact kind of environment we want for reinforcement learning: in order to maximize difficulty, the agent should be trained only from its own visual input. For example, the agent does not "know" that the *attack* action kills zombies; it learns this through training.

[**Hiro: do you want to discuss Minecraft/Malmo implementation issues here? Like the lag issue?**]

4 Method

4.1 Scenario

The agent begins in the middle of a completely flat room with a fixed number of randomly generated zombies. The room has a uniform color so that the algorithm will not have to account for differences in lighting and colors. The zombies automatically approach the agent and damage it if they get too close. The agent will be equipped with a sword to fend off the zombies. As the agent destroys zombies, new zombies continue to spawn (so that there is always a constant number of zombies), and the game continues to run until the character loses all of its health. At this time, the score will be calculated based on the total number of zombies that the character was able to kill and the scenario resets to the beginning.

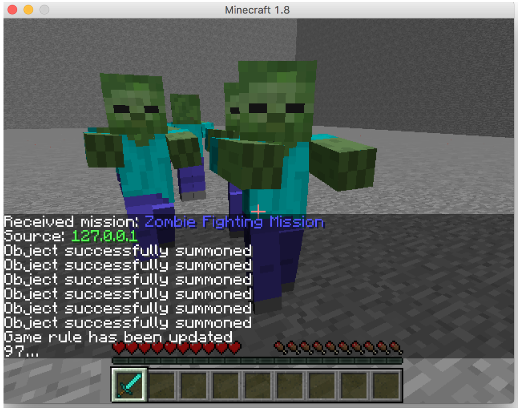
4.2 MDP Formulation

4.2.1 Actions and States

The actions for this problem are the discrete movements of the agent (the agent can move forward, back, left, or right, or remain stationary) as well as the ability to attack with a sword. Minecraft provides the agent with continuous movement actions, but for the purposes of this project we have discretized this into five possible actions.

The states are composed of the image of the player's vision represented by the pixels of the screen. Project Malmo provides the functionality to retrieve the image from the screen and feed this back into our algorithm. In this way, the agent reacts to its environment based on visual input.

Figure 1: Minecraft screenshot



```
Received mission: Zombie Fighting Mission
Source: 127.0.0.1
Object successfully summoned
Object successfully summoned
Object successfully summoned
Object successfully summoned
Object successfully summoned
Object successfully summoned
Game rule has been updated
97...
```

4.2.2 Reward

We assign the algorithm a reward of +1 for a successful hit onto a monster and a reward of +5 for a successful kill, as well as a reward of **???** for every frame for which it stays alive. There is also a negative reward of -1 for every time the agent gets hit.

4.2.3 Deep Q-Learning and Theory

We use Q-learning to have our agent learn a policy to best fight these monsters. We utilize the work done by Google Deep Mind to use a convolutional neural network to approximate the Q-function iteration after iteration.

Algorithm 1: Adapted from Matiisen, 2015

```
initialize Q function with random weights ;
observe initial state $S$ ;
repeat
    with probability $\epsilon$ generate random action ;
    otherwise select $a = \arg\max_{a'} Q(s, a')$ ;
    carry out action $a$ ;
    observe reward $r$ and new state $s'$ ;
    update $D$ with new data: $< s, a, r, s' >$ ;
    sample random transitions from $D$ ;
    update network ;
until terminated;
```

5 Results and Analysis

The agent kicks ass and takes names like Chuck Norris on cocaine.

Table 1: Learning rate under varying Reward Structures

Table 2: Learning rate under varying ϵ

Table 3: Learning rate under varying learning rates

Table 4: Learning rate under varying difficulties

We ran into the unexpected problem that...

6 Conclusions and Future work

We succesfully trained a Deep Q Network to move around a room and kill zombies by taking visual input from the agent. Our results suggest that foo and bar methods worked best because [...]

Future work on this task might focus on expanding the agent's action space so that the agent can execute continuous movements rather than the discretized actions we have programmed. Additionally, researchers could focus on fine-tuning the convolutional neural network to

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*