Jeffrey Barratt

jbarratt@
cs.stanford.edu

Deep Reinforcement Learning for Playing Real Time Strategy Games

Chuanbo Pan

chuanbo@
cs.stanford.edu

Motivation

Competitive Computer Games, despite recent progress in the area, still remain a largely unexplored application of Machine Learning (ML), Artificial Intelligence, and Computer Vision. Real Time Strategy (RTS) games such as StarCraft II provide an ideal testing environment for AI and ML techniques, as:

- They run in **real time** and involve **incomplete information** (the player cannot see the whole battlefield at once).
- Users must **balance** making units, controlling those units, executing a strategy, and hiding information from their enemy to successfully win a game of StarCraft II.
- Simple actions made early on in the game can greatly impact later stages of the game.

Problem Definition

1) The stalker is ordered to attack the zealot

2) The zealot runs at the stalker after being attacked at range

4) The stalker repeats the process by attacking the zealot again

3) The faster stalker can run backwards to remain a safe distance

We focus our efforts on one minigame which involves “kiting”, a process of engaging a short-ranged unit with a longer-ranged unit, where the longer-ranged unit attacks, moves back to a safe distance, then attacks again, repeating the process. In this specific example, it’s possible to take no damage on the Stalker (the long-ranged unit) from the Zealot (the short-ranged unit). We use Q-Learning to learn the game and the PySC2 library get access to game data.

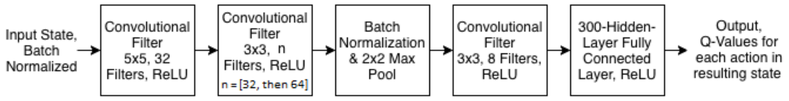
Challenges

Apart from the reasons mentioned in the motivation, StarCraft II is a challenging game because:

- The action space is huge – **O(n^4)** possible selections, and **O(n^2)** possible places to move to.
- The game is in real-time – Can’t spend forever thinking!
- The game is **huge** and **multi-layered**
  - Have to consider attacking, grouping, gathering, etc.
  - Our minigame already includes tasks such as moving to beacon (enemy), attacking, and moving back.
- Q-Learning is difficult with such a huge action/state space

Approaches

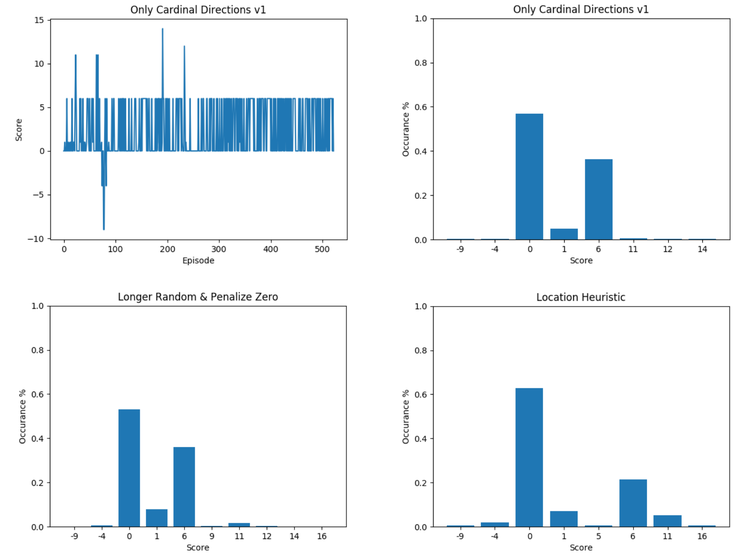
Our approach utilizes Deep Q-Learning. We take advantage of the **image-like** nature of the state by using a Convolutional Neural Network (CNN).



The network outputs the Q-Values for every given state so we don’t have to perform multiple passes. To make Q-Value calculation possible, we also apply a number of techniques as shown in the table below. The *Discretization and Action Limitation* technique is always used. Everything else can be tested in an on-off method.

Technique	Explanation
Discretization & Limiting Actions	• Focus only on Move, Attack, No-Op • Limit agent to move/attack in the four cardinal directions (9 actions total)
Location Heuristic	• Finding the agent can be inefficient • Heuristic assumes the agent made it to the location it was directed to
Penalize ‘Bad’ Actions	• Penalizes each (s, a, r) that results in a score of 0 (assign -1).
Increase Random Time	• Increase the random play time at start • Algorithm might get good starting info

Preliminary Results



Top-left shows training ‘curve’. All other graphs evaluate the network by plotting the fractional of occurrence of every score.

Analysis and Future Work

We can observe several things from our results:

- Theoretical score ranges from -9 to 800. Obviously, we are not near maximum potential.
- The learning curve indicates that we are getting a score of 6 more frequently but still aren’t learning too successfully.
- The actor kites a little but mostly engages in direct combat.
- Vanilla cardinal directions and penalization are similar.
- Location heuristic fails as there are many times where the agent doesn’t actually move from it’s previous location.

Future work on the coming weeks will be focused on:

- Using a LSTM Recurrent Neural Network (RNN) to help determine actions, especially in sequence.
- Refining our abstraction to ensure stability during training
- Reevaluate experience replay and our technique for gathering (s, a, r, s’) to ensure we’re learning correctly.