Machine Learning Engineer Nanodegree

Capstone Proposal

Ji Ma August 12nd, 2017

Proposal

Domain Background

Recently, OpenAI's Artificial Intelligence beat the best player in popular strategy game on 1v1 match, stands for a oustanding milestone after the legend of Deepmind's Alpha Go. It stands for a new era of the AI powered bot in the game way more complex than Go. Also, what interested me so much is that Deepmind also released a brand new Reinforcement Learning Environment for the game StarCraft II, which is one of my favorate game. But the reality is I'm not very good at competing with human players in StarCraft II. Since I am a big fan of Deep Learning and Neural Network, I decided to use the Deep RL to implement the AI to achieve some RL tasks in this brand new handy environment by myself :)

Problem Statement

As declared in the paper along with the Learning Environment, the current best result worked by the most professional researchers from Deepmind could not beat a simple bot in Easy level. It takes so much work and computing powers to achieve the goals like that considering the complexity of the StarCraft II. So my goal should be realistic, tackle the Mini Game like MoveToBeacon and FindAndDefeatZerglings should be a good practice to get started.
Among all those mini games, I will specificly tackle **BuildMarines**, which described below according to pysc2 documents:

Description

A map with 12 SCVs, 1 Command Center, and 8 Mineral Fields. Rewards are earned by building Marines. This is accomplished by using SCVs to collect minerals, which are used to build Supply Depots and Barracks, which can then build Marines.

Initial State

12 SCVs beside the Command Center (unselected) 1 Command Center at a fixed location 8 Mineral Fields at fixed locations Player Resources: 50 Minerals, 0 Vespene, 12/15 Supply

Rewards

Reward total is equal to the total number of Marines built

End Condition

Time elapsed

Time Limit

900 seconds

Additional Notes

- Fog of War disabled
- No camera movement required (single-screen)
- This is the only map in the set that explicitly limits the available actions of the units to disallow actions which are not pertinent to the goal of the map. Actions that are not required for building Marines have been removed.

Datasets and Inputs

Thankfully, Deepmind and Blizzard released the replay dataset which consists of around 60000 entries of replay data from professional players around world. According to Blizzard in their sc2le paper "StarCraft II provides the opportunity to collect and learn from a large and growing set of human replays. Whereas there has been no central and standardised mechanism for collecting replays for StarCraft I, large numbers of anonymised StarCraft II games are readily available via Blizzard's online 1v1 ladder. As well, more games will be added to this set on a regular basis as a relatively stable player pool plays new games."

This dataset could contributes a lot to the supervised learning and deep learning methods. According to Blizzard, "Among professional players it is standard practice to review and analyse every game they play, even when they win."Even the human top players have to learn from the replays. Also, this dataset will benefits the RL process to based on the statement "Learning from replays should be useful to bootstrap or complement reinforcement learning."

Accordong to sc2le, Here are some features of this datasets:

1. The skill level of players, measured by the Match Making Rating (MMR), varies from casual gamer, to high-end amateur, on through to professionals

2. The average number of Actions Per Minute (APM) is 153, and mean MMR is 3789
3. Less than one percent are Masters level replays from top players.
4. Overall, the action distribution has a heavy tail with a few commonly used actions, (e.g., move camera, select rectangle, attack screen) and a large number of actions that are used infrequently (e.g., building an engineering bay).

And here's some example of attributes we could get from the replay data based on pysc2,

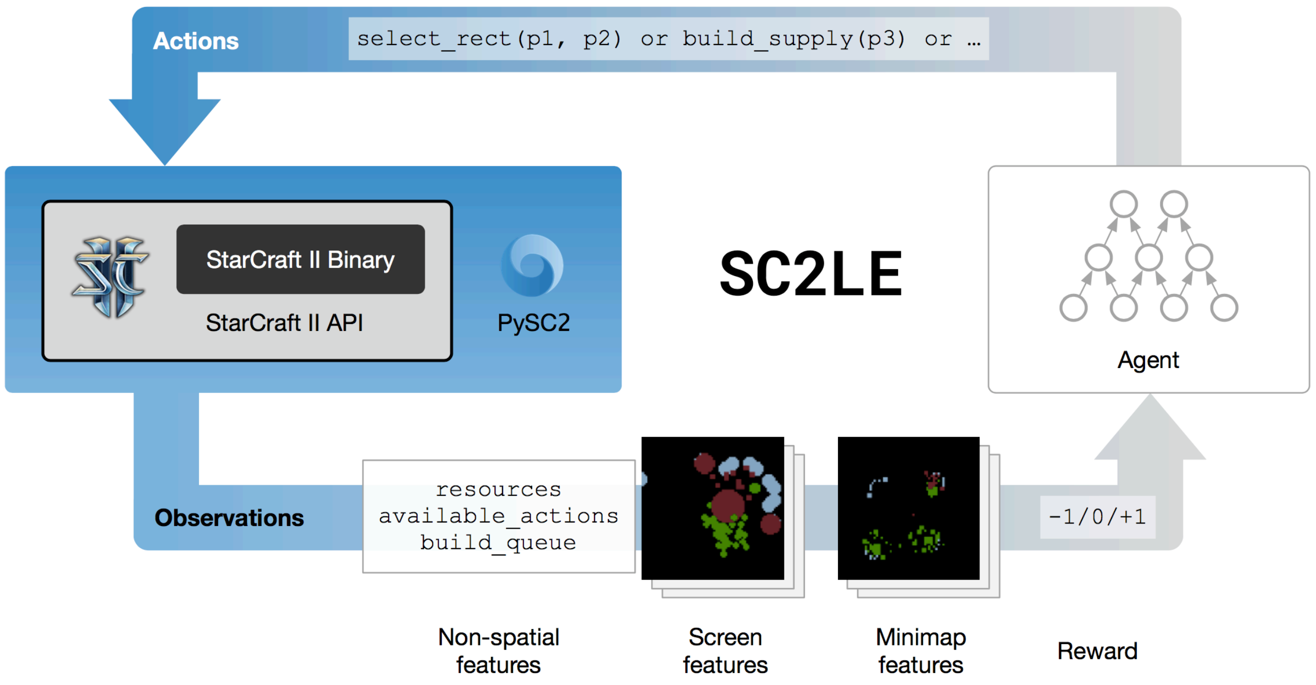
- height_map: Shows the terrain levels.
- visibility: Which part of the map are hidden, have been seen or are currently visible.
- creep: Which parts have zerg creep.
- camera: Which part of the map are visible in the screen layers.
- player_id: Who owns the units, with absolute ids.
- player_relative: Which units are friendly vs hostile. Takes values in [0, 4], denoting [background, self, ally, neutral, enemy] units respectively.
- selected: Which units are selected.

for full features and arttibutes, visit https://github.com/deepmind/pysc2/blob/master/docs/environment.md

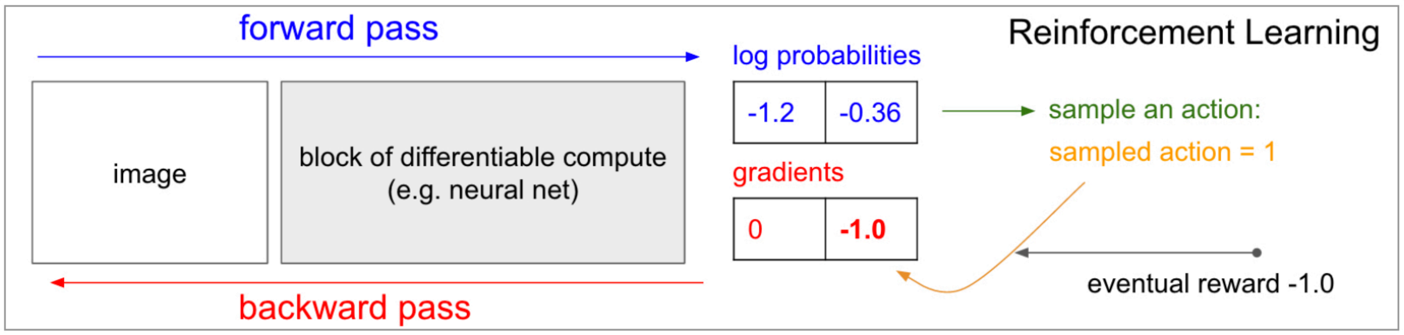
The data is majorly for supervised learning.

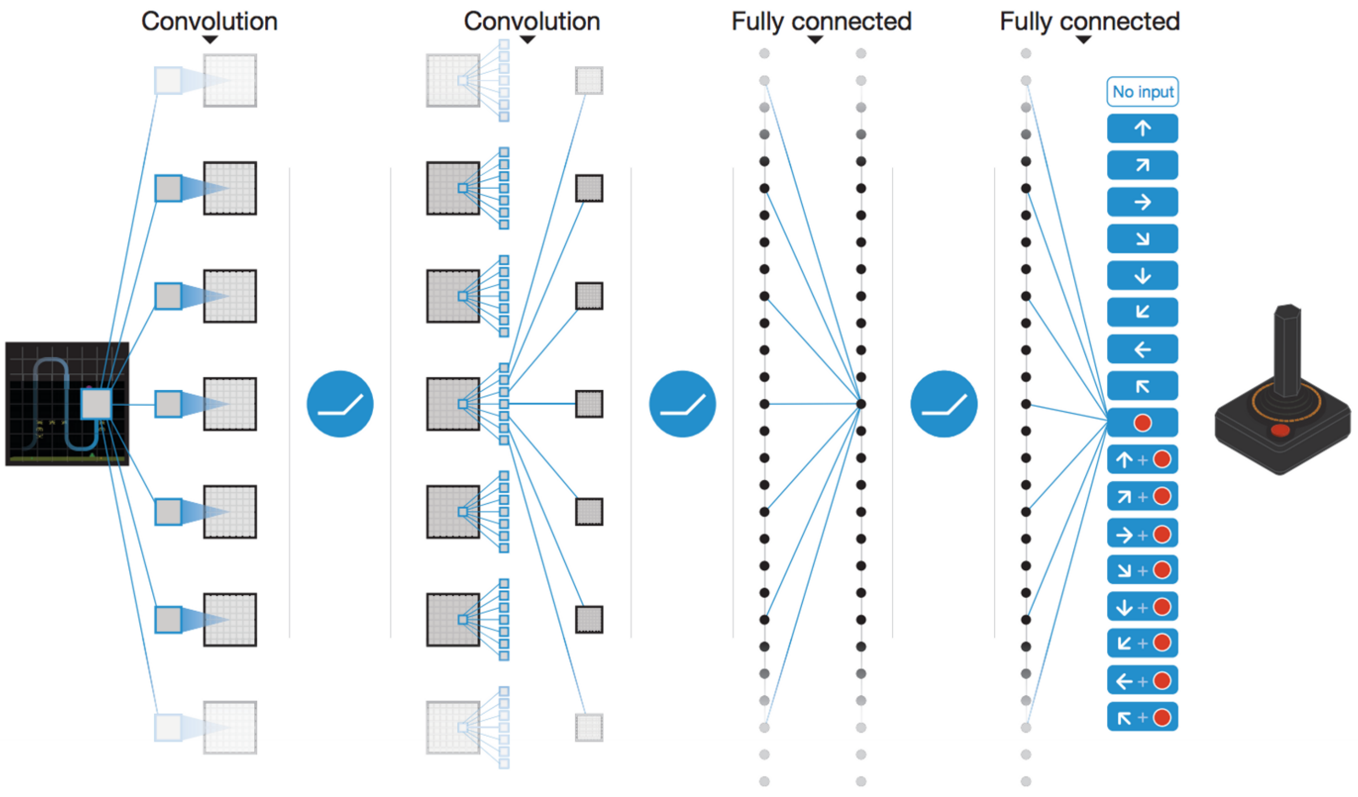
For Deep Q Network, It is all about trial and error, so the dataset described here is not for DQN.

Solution Statement



Deep Q Network





from Q learnig to DQN:

1. Going from a single-layer network to a multi-layer convolutional network.
2. Implementing Experience Replay, which will allow our network to train itself using stored memories from it’s experience.
3. Utilizing a second “target” network, which we will use to compute target Q-values during our updates.

Bellman Equation:
$Q(s,a) = r + \gamma(\max(Q(s',a'))$

loss function
$\text{Loss} = \sum(\text{Q-target} - Q)^2$

Double DQN
$\text{Q-Target} = r + \gamma Q(s',\text{argmax}(Q(s',a,\Theta),\Theta'))$

Supervised Learning: learn both a **value function** (i.e., predicting the winner of the game from game observations), and a **policy** (i.e., predicting the action taken from game observations).

The final solution will be determined along the process of research and trials.

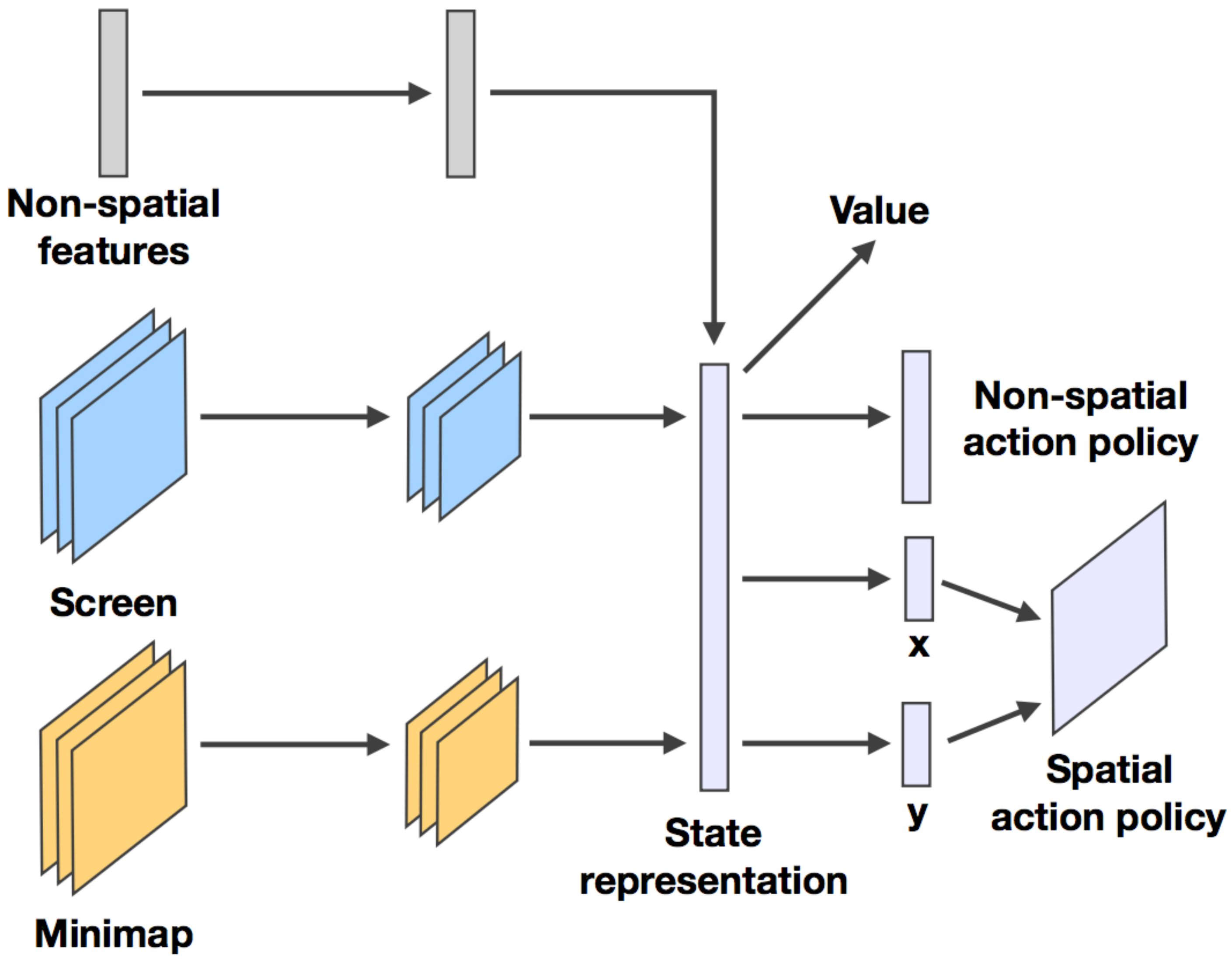
Benchmark Model

Random Agents

According to sc2le paper, they suggested two main random agents.
Random Policy will uniformly pick random action among all valid actions. (Samples in action space)
Random Search will take many independent, randomly policy networks, evaluating each for 20 episodes and keeping the max mean score.(Samples in policy space)

Atari-net Agent

According to sc2le paper, the Atari-net agent is a close adaption of the architecture successfully used for the Atari benchmark and DeepMind Lab environments.



(a) Atari-net

Evaluation Metrics

Reward total is equal to the total number of Marines built。

Project Design

1. Go over all part of Deep RL tutorial provided by Arthur Juliani. Get familiar with Deep RL with Tensorflow.
    1. Deep Q Learning
    2. Agent Visualization
    3. Q-Learning Agents
    4. etc

2. Research on related papers include sc2le, Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games, etc.
3. Get familiar with pysc2 environment, read through all documents include
    1. environment
    2. maps
    3. mini_games

4. Get hands on with replay data, preprocessing, and analysis.
5. Implement BuildMarines described in sc2le from scratch, compare to the result present in paper.
    1. Deep RL
    2. Supervised Learning from replays

6. Create mini games like how to Micro Control Reaper, how to effectively use Seige Tank, etc.(Maybe)
7. Implement Agents
8. Benchmarking, modifying and toning hyper parameters.

References

github.com/Blizzard/s2client-proto
github.com/deepmind/pysc2
Simple Reinforcement Learning with Tensorflow series by Arthur Juliani
Deep Reinforcement Learning: Pong from Pixels by Andrej Karpathy