Machine Learning Engineer Nanodegree

Capstone Project

Ji Ma
August 23st, 2017

I. Definition

(approx. 1-2 pages)

Project Overview

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.

Metrics

Reward total is equal to the total number of Marines built

II. Analysis

(approx. 2-4 pages)

Data Exploration

The data I will first explore around is pysc2 itself.
According to it's source code:

Action space

The action space is huge which is 524 actions for the regular game.

We could get valid actions dynamicly in each step

Status Space

Depends on unit type, the status space could be enormous

1. screen*xy = 84 (the screen size will be screen*xy* screen_xy) we could change it later
2. unit*type*size = 1850 (The unit type size)
3. so, the status space would be screen*xy^2 * unit*type_size

In each Step in Deep RL, pysc2 provided a handy `TimeStep` class which will provide 4 major features for Deep RL:

- step_type: First, Mid, Last(useful to find the step progress in the episode)
- Reward
- Discount: the discound initialized in env
- Obervation: infomation contains
  - screen(different type), unit_type screen will be used
  - minimap
  - etc

In order to know the performance of Random Agent and in the mean while get a sense of data generated in following aspects:

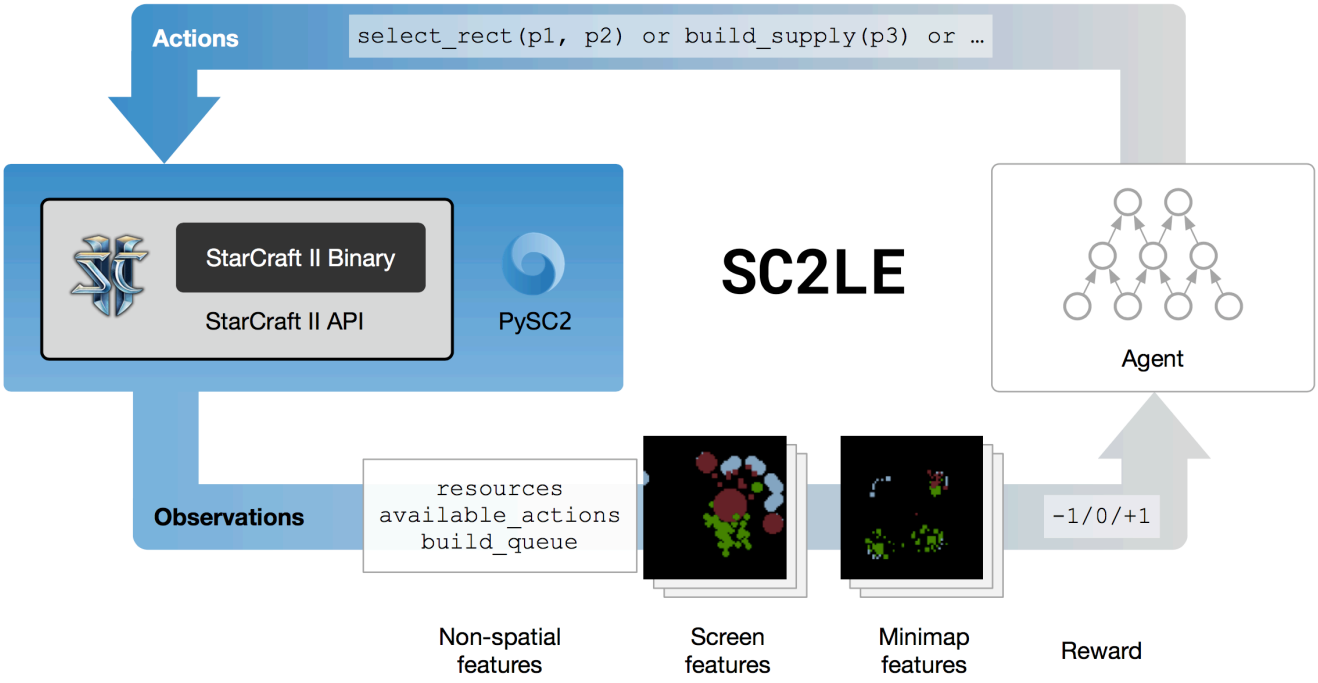
- screenList: the numpy array, will help me narrow down the unit type for minigame
- actionList: the numpy array, will help me narrow down the action space for minigame
- rList: the overall reward(performance matric)

Each of the Build Marine mini game has 15000 frames limit, and I used step_mul=8 as default(8 frames per action).
So, 15000 frames means 1875 steps per game.
I want 4000 games/episode sample data here, so I use 1875000 steps on 4 threads to collect data
`python test_build_marine.py --map BuildMarines --max_agent_steps 1875000 --parallel 4 --norender --nosave_replay > log.dat`

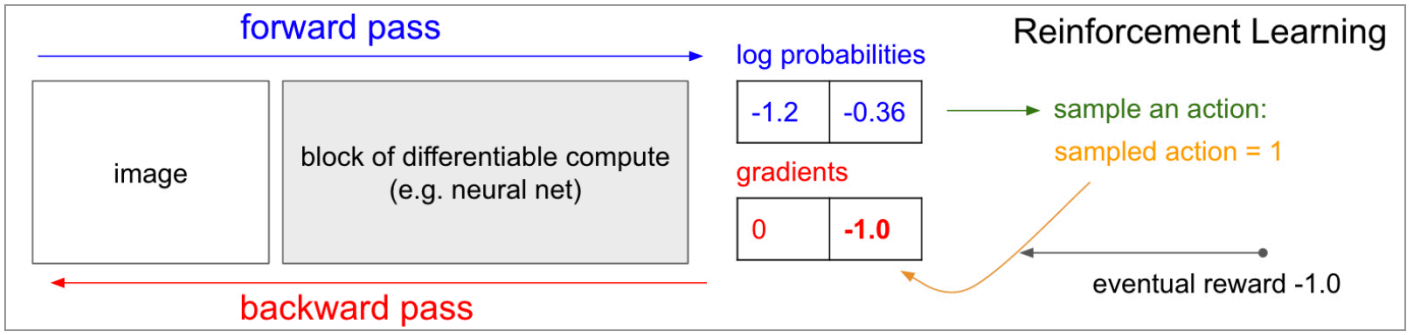
Exploratory Visualization

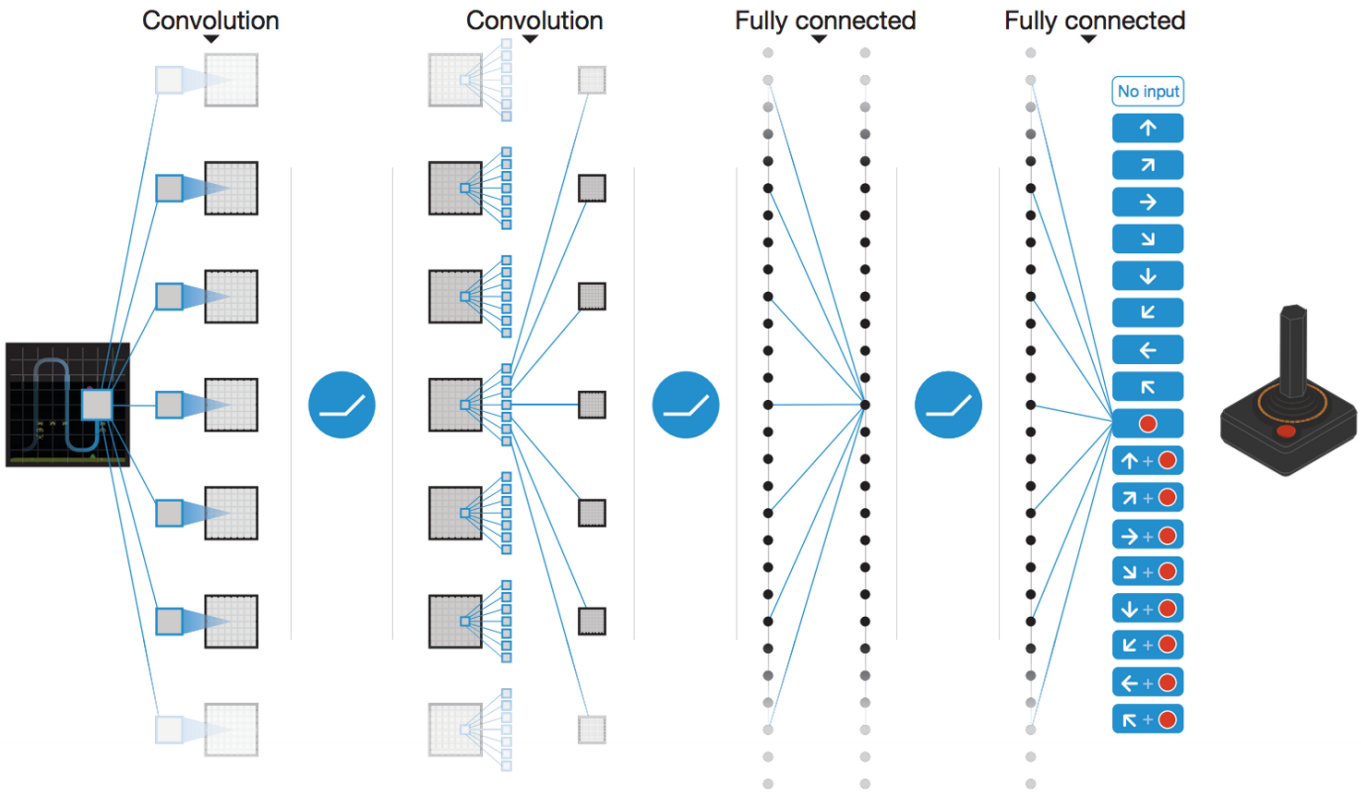
In this section, you will need to provide some form of visualization that summarizes or extracts a relevant characteristic or feature about the data. The visualization should adequately support the data being used. Discuss why this visualization was chosen and how it is relevant. Questions to ask yourself when writing this section: - *Have you visualized a relevant characteristic or feature about the dataset or input data? - Is the visualization thoroughly analyzed and discussed? - If a plot is provided, are the axes, title, and datum clearly defined?*

Algorithms and Techniques



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

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).

Benchmark

Random Agents

According to sc2le paper, they suggested two main random agents.
I will particularly use **Random Policy**
Random Policy will uniformly pick random action among all valid actions. (Samples in action space)

III. Methodology

(approx. 3-5 pages)

Data Preprocessing

In walkthrough ipython notebook.

I derived the overall valid unit_type input and action input.

In this section, all of your preprocessing steps will need to be clearly documented, if any were necessary. From the previous section, any of the abnormalities or characteristics that you identified about the dataset will be addressed and corrected here. Questions to ask yourself when writing this section: - *If the algorithms chosen require preprocessing steps like feature selection or feature transformations, have they been properly documented? - Based on the **Data Exploration** section, if there were abnormalities or characteristics that needed to be addressed, have they been properly corrected? - If no preprocessing is needed, has it been made clear why?*

Implementation

Deep Q-Learning Algorithm

```
initialize replay memory D
initialize action-value function Q with random weights
observe initial state s
repeat
    select an action a
        with probability ϵ select a random action
        otherwise select a = argmaxa'Q(s,a')
    carry out action a
    observe reward r and new state s'
    store experience <s, a, r, s'> in replay memory D

    sample random transitions <ss, aa, rr, ss'> from replay memory D
    calculate target for each minibatch transition
        if ss' is terminal state then tt = rr
        otherwise tt = rr + γmaxa'Q(ss', aa')
    train the Q network using (tt - Q(ss, aa))^2 as loss

    s = s'
until terminated
```

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section: - *Is it made clear how the algorithms and techniques were implemented with the given datasets or input data? - Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution? - Was there any part of the coding process (e.g., writing complicated functions) that should be documented?*

Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section: - *Has an initial solution been found and clearly reported? - Is the process of improvement clearly documented, such as what techniques were used? - Are intermediate and final solutions clearly reported as the process is improved?*

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model's solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section: - *Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate? - Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data? - Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results? - Can results found from the model be trusted?*

Justification

In this section, your model's final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section: - *Are the final results found stronger than the benchmark result reported earlier? - Have you thoroughly analyzed and discussed the final solution? - Is the final solution significant enough to have solved the problem?*

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section: - *Have you visualized a relevant or important quality about the problem, dataset, input data, or results? - Is the visualization thoroughly analyzed and discussed? - If a plot is provided, are the axes, title, and datum clearly defined?*

Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section: - *Have you thoroughly summarized the entire process you used for this project? - Were there any interesting aspects of the project? - Were there any difficult aspects of the project? - Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?*

Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section: - *Are there further improvements that could be made on the algorithms or techniques you used in this project? - Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how? - If you used your final solution as the new benchmark, do you think an even better solution exists?*

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 http://neuro.cs.ut.ee/demystifying-deep-reinforcement-learning/

Before submitting, ask yourself. . .

- Does the project report you've written follow a well-organized structure similar to that of the project template?
- Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
- Would the intended audience of your project be able to understand your analysis, methods, and results?
- Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
- Are all the resources used for this project correctly cited and referenced?
- Is the code that implements your solution easily readable and properly commented?
- Does the code execute without error and produce results similar to those reported?