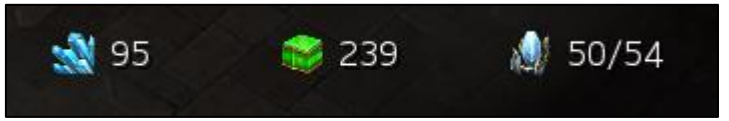
An entry-level StarCraft II AI program

ZhongMengyan 2019533233

Introduction

- My project is to combine the knowledge of artificial intelligence and machine learning to write an AI program about the RTS game *StarCraft 2.*
- Language: Python

Issues to consider



Minerals Vespene Gas Population

* The entire Project selects only Protoss.

- Economy
  - Resources: Minerals & Vespene Gas
  - Number of “Nexus” (fundamental building for Protoss)
  - Number of “Probe” (Protoss worker unit, harvests resources)
- Population ratio
  - Population cap is 200.
  - Too many “Probe” will result in a small army.
  - Too few “Probe” will result in a slow speed to harvest resources.

Probe

Build

Nexus

Issues to consider

- Scout
  - The focus of RTS games.
  - Provide information about enemy units to help AI choose actions.
  - Mainly use “Observer” (spy drone for Protoss) to scout.
- Military
  - Number of “Gateway” (one of barracks for Protoss) and other similar buildings.
  - The ratio of different combat units.
  - Strategy: When to attack & defend.

Observer

Stalker

Build

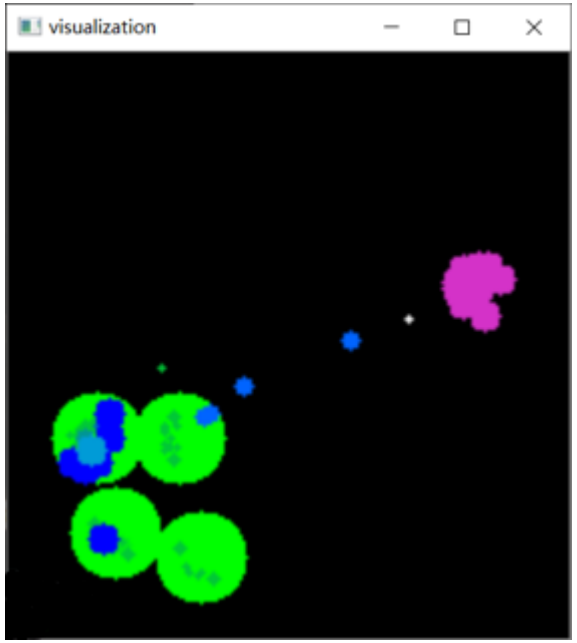
Gateway

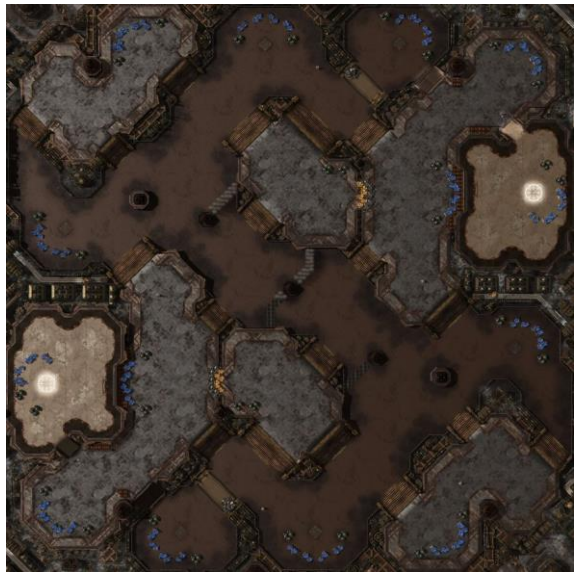
Building training data

* The complexity of the game: the number of variables is also variable.

- Data
  - Should be a reflection of the battlefield situation at a certain moment.
  - Visualization: Display own units and enemy units (in different colors) in Opencv.

  * So the visualization and neural networks are only suitable for a single map.





AutomatonLE
Ladder 2019 Season1
Size: 148x148

Building training data

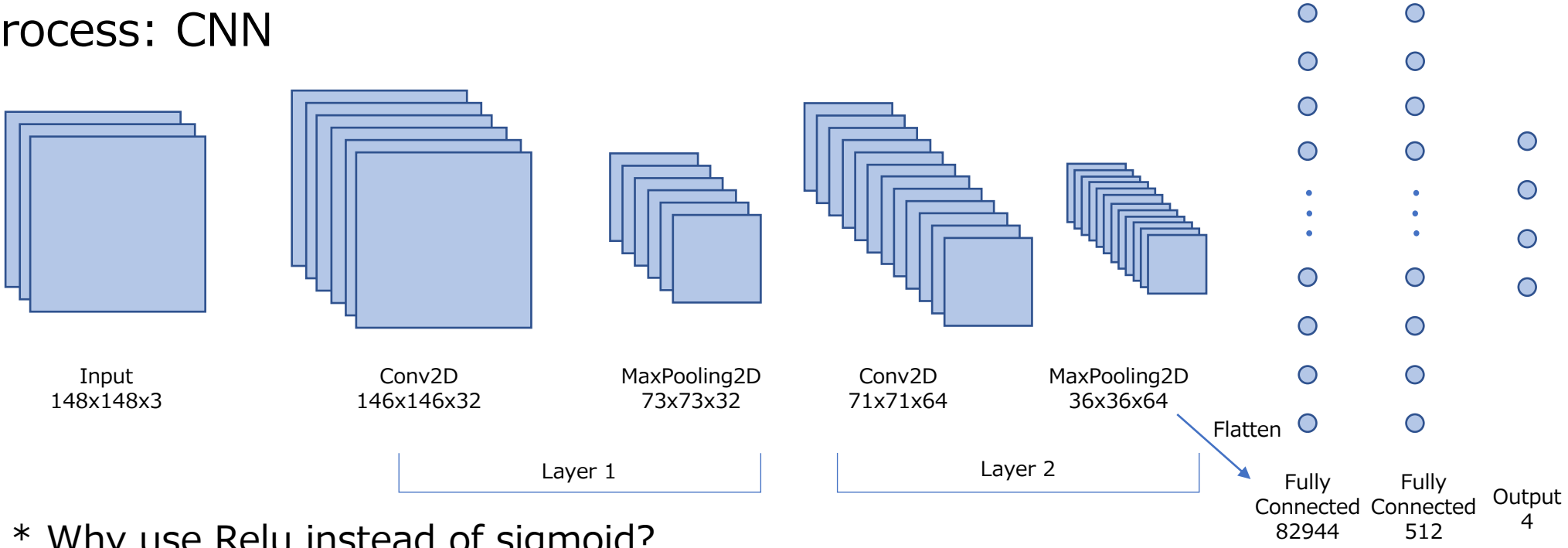
- Label
  - In the above situation, what action should be taken?
  - A simple consideration is to pre-arrange a number of actions. For example:
    1. Attack any visible enemy unit
    2. Go to a random point out of view (scout)
    3. Stay (do not do anything)

```python
@property
def time(self) -> Union[int, float]:
    """ Returns time in seconds, assumes the game is played on 'faster' """
    return self.state.game_loop / 22.4  # / (1/1.4) * (1/16)
```

- Time in game
  - Obviously, the program is asynchronous.
  - All async functions are called once every game loop (about 45*ms*).
- Get data
  - At a certain moment, if all combat units are idle, then choose an action randomly.
  * This gets a 148x148x3 matrix, each data in it is type “uint8” (RGB).
  * Its label is a *n*x1 matrix (*n* is the number of actions).
  - After a game, we get a number of data, and store them in an “npy” file.

Building Neural Network Model

- Library: Keras & TensorFlow
- Process: CNN



* Why use Relu instead of sigmoid?

* Why use Dropout?

Evaluation

* Bot only chooses Protoss, and map is AutomatonLE.

- After training data from about 2500 games:
- Randomly choose

Win Rate (%)	Vs. Terran	Vs. Protoss	Vs. Zerg
Easy	96.5	100.0	86.7
Normal	45.3	40.3	61.2
Hard	0.0	9.5	25.0

- Use Model

Win Rate (%)	Vs. Terran	Vs. Protoss	Vs. Zerg
Easy	100.0	100.0	97.6
Normal	81.0	78.9	86.8
Hard	21.4	33.8	40.2

Follow-up work

- Improve actions
  - Increase the number of actions.
  - Add other types of actions (Now only actions of combat units).
- Improve visualization
  - As input, visualization is very important for model training.
  - Adjust parameters (like circle size of a certain type unit, or color)
- Micro manipulation
  - There is no limit to APM. The average APM of top professional players is in the hundreds, while AI can easily reach tens of thousands.
  - So AI can easily do some things unimaginable by human players, for example, finely manipulate every unit.

Thanks!