Basic Deep Q-learning Bot for FightingICE AI platform

This is a simple description of an example Deep Q-learning (DQN[1]) AI, BasicBot, for fighting game AI platform FightingICE[2] We have implemented a simple Deep Q-learning algorithm with replay memory in Python 3.5. BasicBot collects a state and action pair through game playing and stores it into replay memory. To speed up this process, we run multiple games in parallel. Simultaneously, the main thread of the training program updates Q-function (neural network) using collected data. The neural network has two convolution layers and two fully connected layers. It determines which action is the best action based on game screen input and its current energy value. The network was trained using tensorflow and converted for non-tensorflow environment using tfdeploy.

We trained this bot against DisplayInfo Bot (simple scripted bot). Although we could not show outperforming performance against DisplayInfo, BasicBot showed some proper behavior in each situation. This implies BasicBot could learn to do a simple task.

We launched multiple games in the training phase to increase data gathering speed. Since we could not control game flow freely and skip the unnecessary time consuming for keeping game frame rate, we had to wait for each frame to collect one data tuple (s_1, a, s_2, r). Instead, we could collect data from other games with multiple instances. Even though there are more advanced methods [3] , using multiple instances of a game for reinforcement learning, we used the standard DQN with replay memory. Therefore, there are no significant changes in our method, compared to the original one, except for the data gathering process.

Environment setting

Simple explanation of each file

- BasicBot.py
  - AI implementation
- BasicBot.pkl
  - Pretrained AI (neural network's weights)
- DisplayInfo.py and SandBag.py
  - Opponent AIs for training
- Main_PyAIvsPyAI_BasicBotvcDisplayInfo_6500.py
  - Executes the AIs for a fight

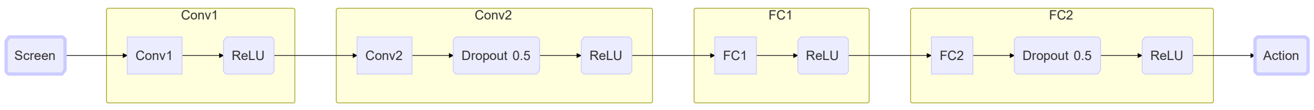
Installation
Copy all files to Python directory in FTG3.10 and copy 'BasicBot.pkl' to '..FightingICEver.3.10/data/aiData/BasicBot/.'

```
─┬ FightingICEver3.10
 └ Python       <-- copy to here
```

or

```
─┬ FightingICEver3.10
 ├ Python
 └ BasicBot       <-- copy to here
```

Neural Network Architecture



- n
    - Number of samples in mini batch
- Screen (Input)
    - Current and previous three gray scaled screen images
    - Resolution: 38×57; higher resolution needs more time to get an image
    - The AI was P1, and its opponent or P2's pixel values were inverted during training. Note that because in the competition all games will be run in "--inverted-player 1", the following lines have been introduced to invert the screen when the AI is P1 (in other words, to ensure the opponent is always inverted): `if self.player:`
      ```
                  screen = -screen
      ```
    - All pixel values are normalized to [-1,1]
- Energy (Input)
    - Divided by 300 to normalize
    - Single float point value
- Conv1
    - 2d convolution layer: # of filter: 8, kernel size: 6, stride: 2
- Conv2:
    - 2d convolution layer: # of filter: 8, kernel size: 3, stride: 1
- FC1:
    - Fully connected layer: # of node 256
- FC2:
    - Fully connected layer: # of node 3
-

- Action (output)
    - Each node's output represents the Q-value of one of the 10 action types below.
    - Actions: key input when BasicBot in the left side (change L and R when BasicBot in the right side; ' ' means no key input)
        - L, , , , (defense)
        - LA, , , , (throw)
        - L, ,L, , , , (backstep)
        - R, ,R, , , , (Dash)
        - B, ,B, , B, , (B)
        - DB, ,DB, ,DB, , (2_B)
        - D, DR, RA, , , , (2 3 6_A)
        - D, DR, RB, , , , (2 3 6_B)
        - D, LD, LA, , , , (2 1 4_A)
        - D, LD, LB, , , , (2 1 4_B)
    - BasicBot chooses the best action that has the highest Q-value and sufficient energy to perform it. Although the action has the highest value, it will be discarded if there is not enough energy to perform it.

Testing

After running the FightingICE server with the options

--py4j --port 6500 --black-bg --inverted-player 1

run the following command:

```
~$  python  Main_PyAIvsPyAI_BasicBotvsDisplayInfo_6500.py
```

It finds 'BasicBot.pkl' (parameter file) from the current working directory or '../FightingICEver.3.10/data/aiData/BasicBot/.'

Note that according to the competition rules, the parameter file must be placed in '../FightingICEver.3.10/data/aiData/BasicBot/.'

---

1. V. Mnih et al., “Playing Atari with Deep Reinforcement Learning,” arXiv:1312.5602 [cs], Dec. 2013. ↩
2. “Welcome to Fighting Game AI Competition.” [Online]. Available: http://www.ice.ci.ritsumei.ac.jp/~ftgaic/. [Accessed: 17-Mar-2017]. ↩
3. V. Mnih et al., “Asynchronous Methods for Deep Reinforcement Learning,” arXiv:1602.01783 [cs], Feb. 2016. ↩