AI Agent for the Battle City

Jiaxing Geng, Junjie Dong, Zihuan Diao
{jg755, junjied, diaozh} @ stanford.edu

Problem Definition

The purpose of this project is to build an AI agent to play the classic game, the Battle City. In this game, the agent's goal is to shoot the enemy tanks and defend the base. The evaluation metric for the agent is the final score.
The agent's action space consists of two dimensions:

- Move Direction: {up, down, left, right, none}
- Shoot: {shoot, none}

Challenges and Approach

Challenges:

- Game setting is complicated, hard to model transitions between states.
- Number of game state is huge.
- Sparse default rewards.

Approach:
This project models the game as a Markov Decision Process (MDP). In order to cope with the challenges, the project uses a reinforcement learning approach with the following techniques:

- **Q– learning**: off-policy. No need to model transitions.
- **Function Approximation**: Generalizes the game states. In this project, we explore three options: a linear model, and two different neural network models.
- **Dropping Frames**: Samples and reacts every 5 frames.
- **Non-trivial Reward Function:** give reward when agent behaves positively.

Linear Model

Features: Duplicates the following with each action, size: 420

- Bullet position, direction
- Enemy position, direction and type
- Proximity map information
- Agent's distance to enemies
- Agent's distance to base
- Enemy's distance to base

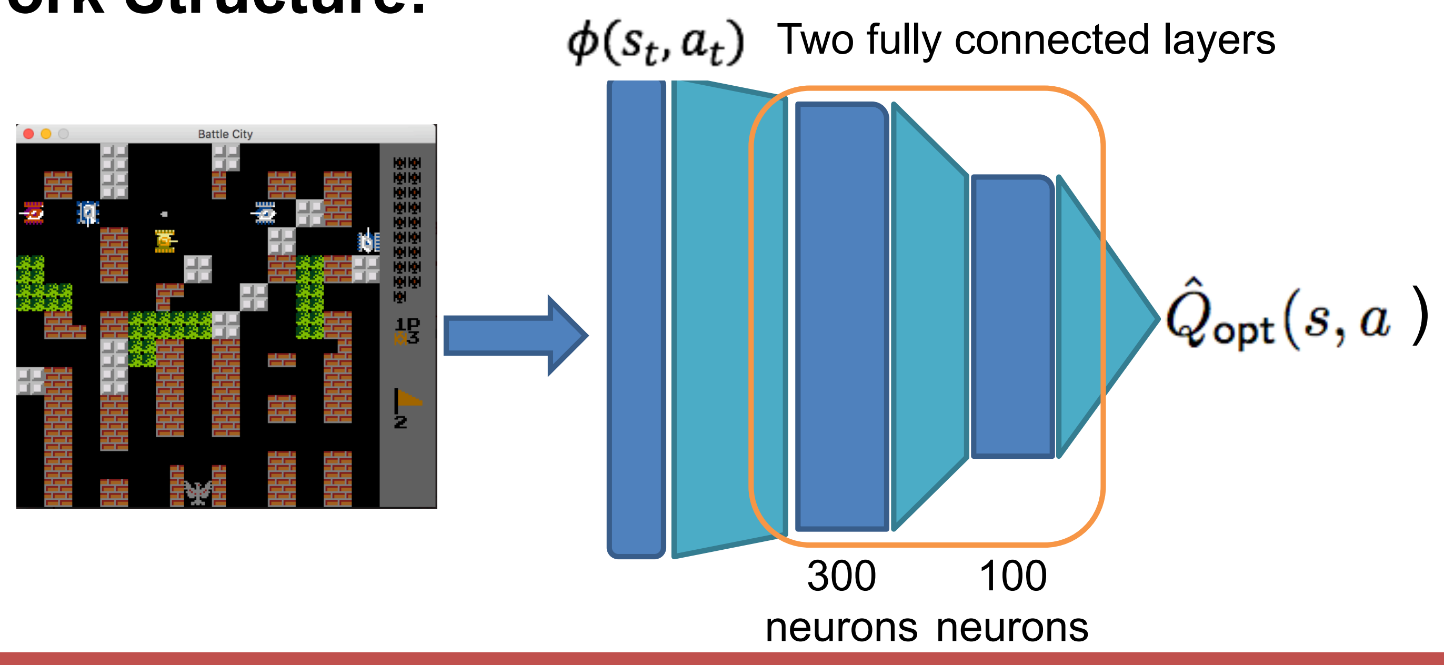
$$\hat{Q}_{\text{opt}}(s, a; \mathbf{w}) = \mathbf{w} \cdot \phi(s, a)$$

Training: Using Replay buffer with mini-batch gradient decent.

$$w \leftarrow w - \eta \cdot \frac{1}{8}\sum_{t=1}^{8}\left([\hat{Q}_{opt}(s_t, a_t;\, w) - (r_t + \gamma \widehat{V_{opt}}(s_t{}'))] \cdot \phi(s_t, a_t) + \lambda w\right)$$

Deep Q-learning v1

Features: The same features used in the linear model.
Neural Network Structure:

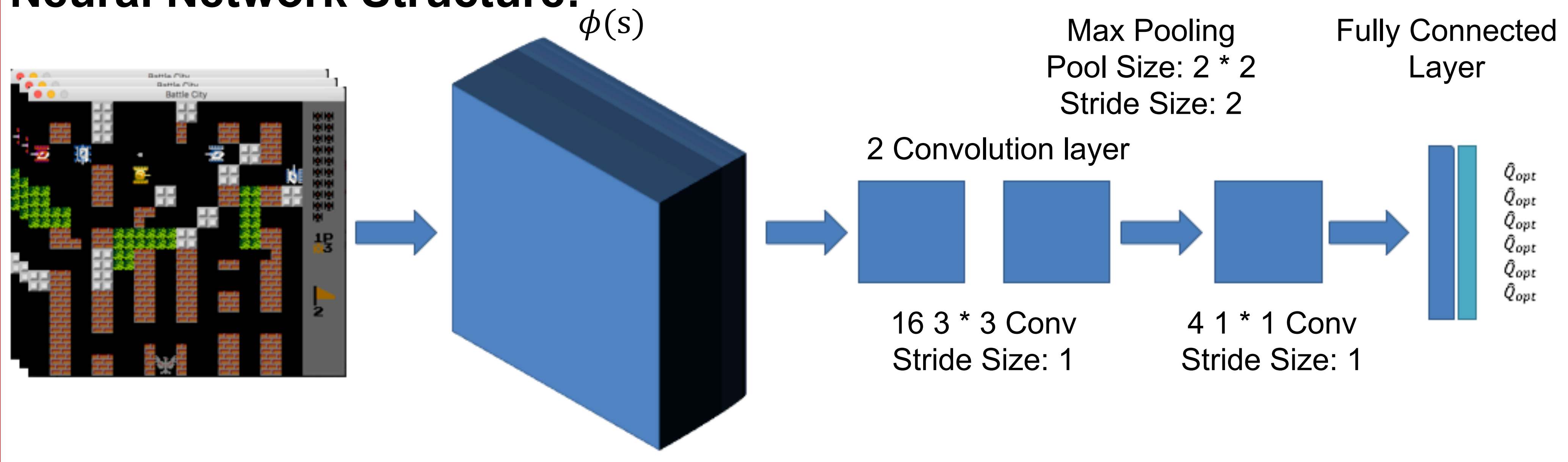


Deep Q-learning v2

Features: Raw map information, size: 26 * 26 * (6 * 3)

- Each frame has 6 layer of map information.
- Record the previous 2 frames.

Neural Network Structure:



Output: Q value for all possible actions in state S.

Result

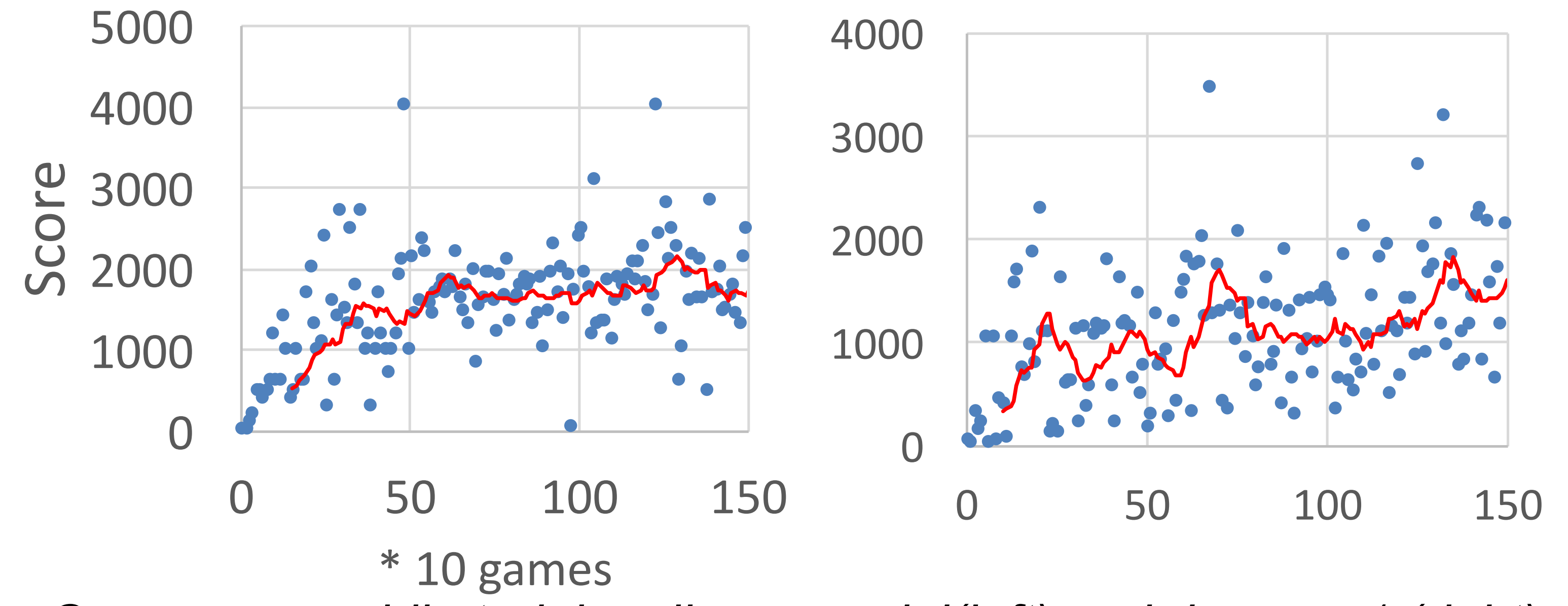
Model	Max Score	Average Score
Baseline - Random	1600	145.7
Oracle - Human	12400	5611.2
Linear (1500 games)	7500	1852.6
Deep Q v1 (1500 games)	3500	1308
Deep Q v2 (450 games)	2300	1000

Game Score Result 100 games

Observations:

- The linear model achieves the best performance.
- All three models have a better performance than the baseline random agent.
- None of them are reaching human oracle level performance.

Analysis



Game score while training, linear model(left) and deep q v1 (right)

- The two deep Q learning model can **perform better with more training**. The game score is increasing as we train the deep Q v1 model.
- The project's models are **not capturing the idea of defending the base.**
- **The agents are behaving less active** than human players, this helps the agents to stay alive and get more points.
- The agents would head into a wall or shoot at indestructible tiles, showing that the **model's action space should be more specific.**

Acknowledgement

We would like to thank Prof. Percy Liang, Prof. Stefano Ermon, and Zach Barnes for their advice and generous help.