Minecraft AI: Finding Water Sources

Lydia Chan[1] and Russell Tran[1], {lchan528, tranrl}@stanford.edu

[1]*CS221 Artificial Intelligence: Principles and Techniques, Stanford University*

Stanford
Computer Science

Overview

Problem: Complete a search task in an unknown environment
Motivation: Autonomous robots for dangerous search and rescue missions

Approach: Find a source of water in a virtual environment in **Minecraft**

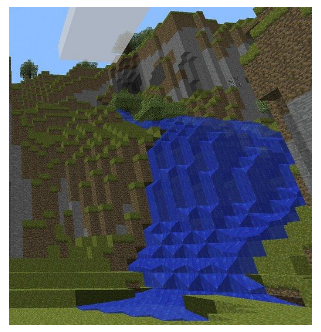

Fig 1. Waterfall in Minecraft

Fig 2. Large Minecraft Lake

Dataset

Dataset Sources

- **MALMO** to design a variety of interactive Minecraft worlds
- **MineRL** to interface with the environments using OpenAI Gym

Dataset Environments

- **Basic**: Rectangular box made of a uniform material (stone)

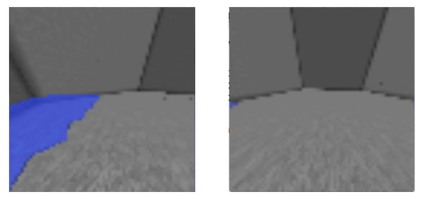

Fig 3. Examples of a Basic Environment

- **Sparse:** flat, grassy terrain

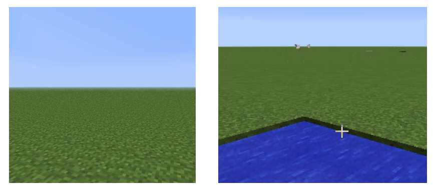

Fig 4. Examples of a Sparse Environment

- **Dense:** hilly, jungle-based terrain with animals, deserts, or tundras

Fig 5. Examples of a Dense Environment

Baseline

Brute-force policy

- Agent ignores all sensory input
- systematically traverses every tile in the environment until it inevitably touches the target
- High standard but fails outside of basic environment

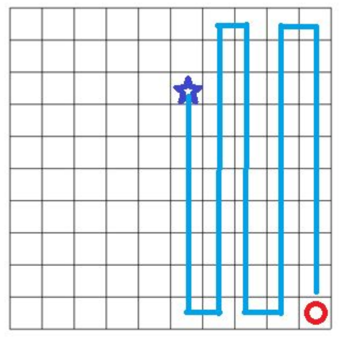

Fig 6. Illustrated Approach of Baseline

Q-learning

Naive approach

- Identity feature extractor for sensory (pixel) input
- Constant exploration rate of 30%

Heuristics-based approach

- Feature extractor captures the value of blue pixel from each pixel in the 64x64 frame

Deep Q-learning (DQN)

- **Comparison with Q-learning**
  - Feature extractor was time-expensive
  - We wanted to evolve the exploration rate
- Feature extractor (sensory input) is based on deep neural networks
- Linear exploration schedule transitions from a randomness exploration rate of 100% to 2% in more than 100,000 time steps
- Stochastic gradient descent was used to train the DQN

Results

*Reward: +6000 for touching water, -1 per step taken

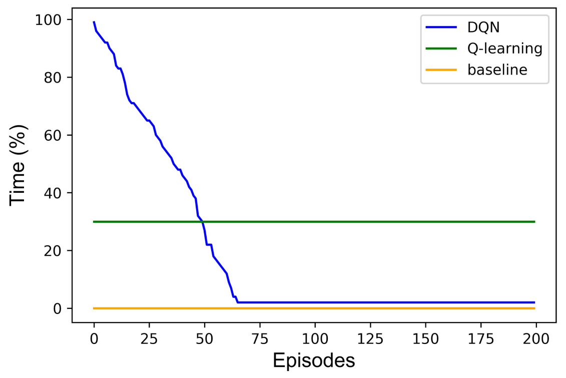

Fig 7. Percentage of Time Spent Exploring per Episode

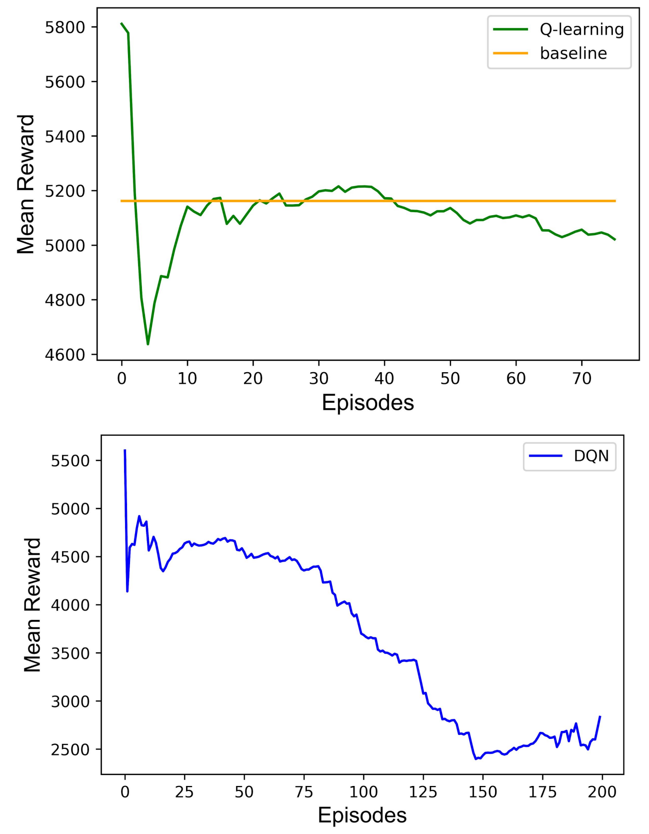

Fig 8. Mean Episode Reward

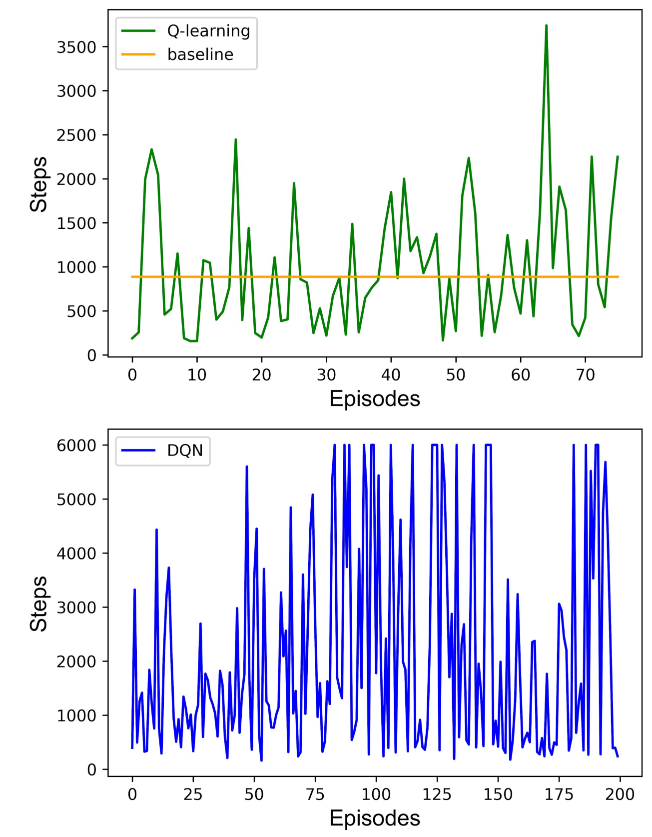

Fig 9. Number of Steps per Episode

Discussion

Memory Constraints

- Naive approach to Q-learning exceeds realistic memory constraints; better features are necessary

Number of Steps per Episode

- No evident correlation: we may reconsider this as a metric, or we need to improve the performance of Q-learning and DQN algorithms

Mean Episode Reward

- Interesting downward trend in DQN, but seems to stabilize, so DQN seems to be converging

DQN did **not** outperform Q-learning and Baseline algorithms, which is consistent with Guss et al. who found DQN only performs better in dense environments.

Future Work

- Sparse and dense environments for more realistic scenarios and compare DQN
- Incorporate experience replay to improve the performance of DQN
- Hyperparameter tuning to optimize general performance

References

- Gray, Jonathan, et al. “CraftAssist: A Framework for Dialogue-Enabled Interactive Agents.” ArXiv.Org, 2019, arxiv.org/abs/1907.08584. Accessed 24 Oct. 2019.
- Guss, William, et al. NeurIPS 2019 Competition: The MineRL Competition on Sample Efficient Reinforcement Learning Using Human Priors. 2019.
- Szlam, Arthur, et al. Why Build an Assistant in Minecraft? 2019.
- Udagawa, Hiroto, et al. Fighting Zombies in Minecraft With Deep Reinforcement Learning. 2016.