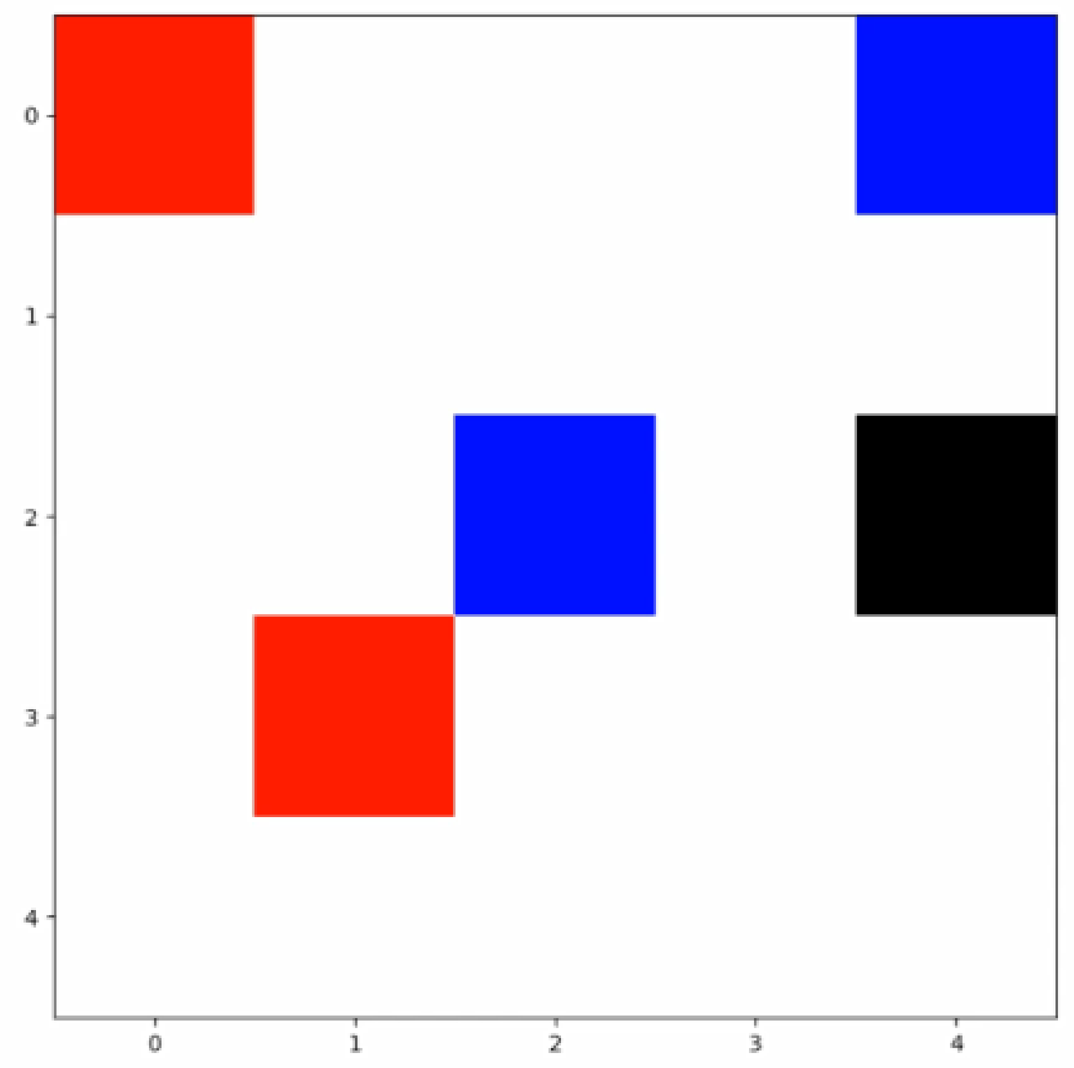
MULTI-AGENT REINFORCEMENT LEARNING ENVIRONMENT (and how I solved it)

Hoan Tran

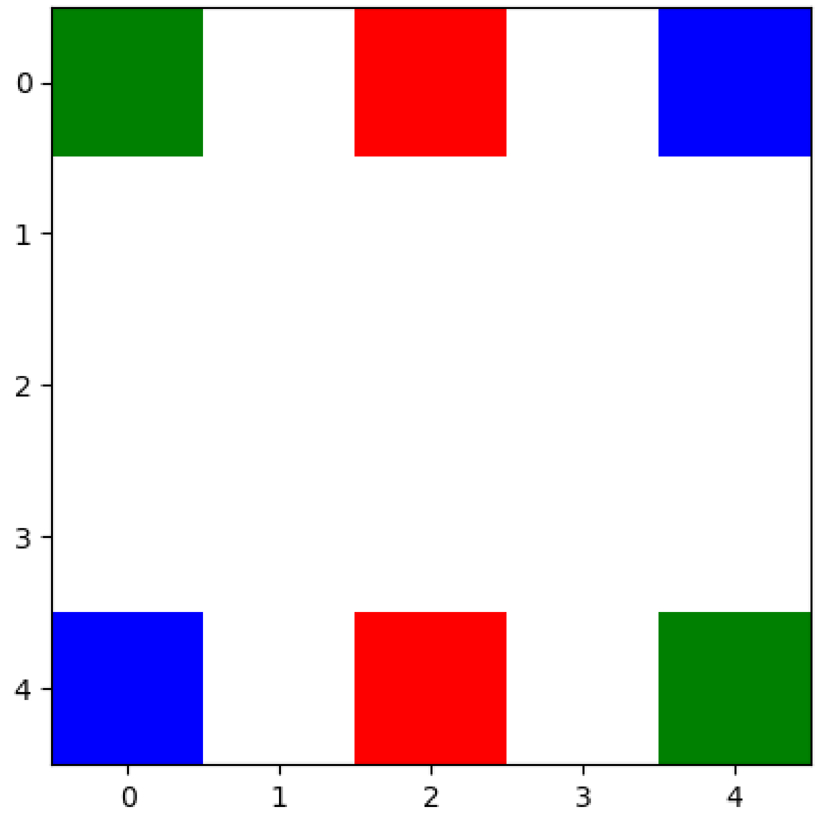
Background

- Mimics real-world situation, where people both compete, and cooperate for common goal.
    - Consider driving: when should we yield?
- Multi agents make the world more complex:
    - The state changes depending on other agents.
    - While learning, the state distribution probabilities is affected by other agents' action, which is also changing rapidly.

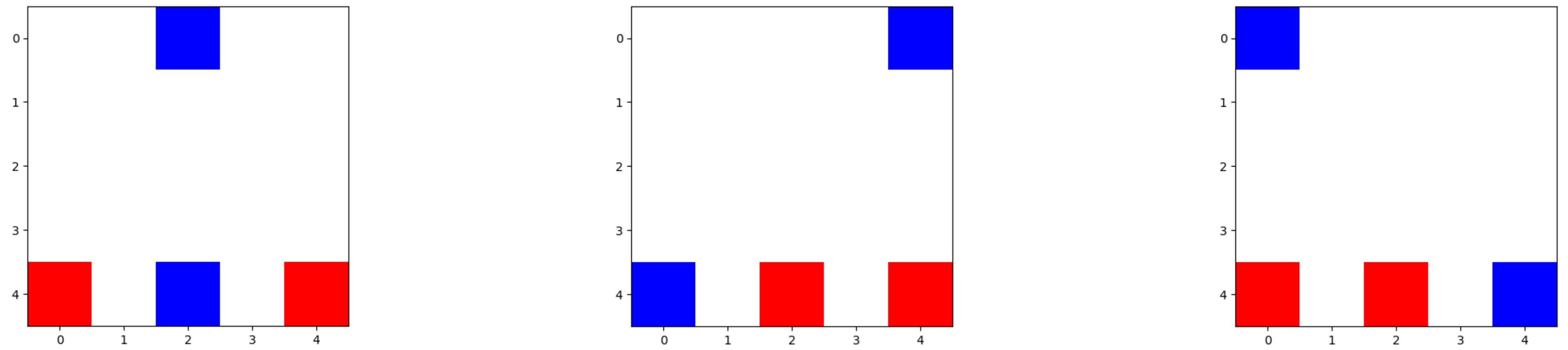


The environment

- Fully-observable, deterministic.
- Custom-defined grid environment
- Agents start at the bottom, and navigate to the top
- The destination are designed so the paths are intersected.
- Following the presentation, we will deal with a blank 5x5 grid and 3 agents.



An example of a 5x5 grid with 3 agents.



Rewards, Observation and Action Space

- No-OP, Left, Right, Up, Down
- Can observe the grid, the current locations of all agents, and the target.
- Independent reward:
  - -1 at every steps, except at the target.
  - -5 for illegal move.
  - 0 at every steps, when reached the goal (while also cannot move)

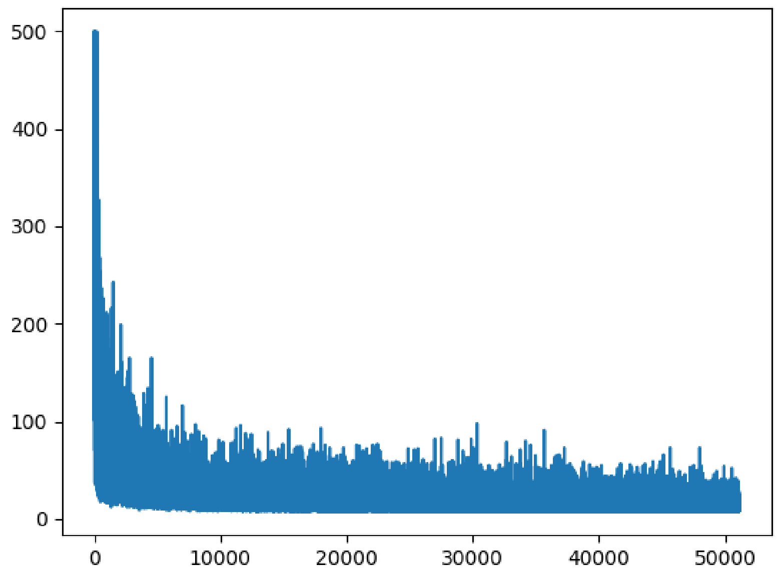
Q-Learning

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

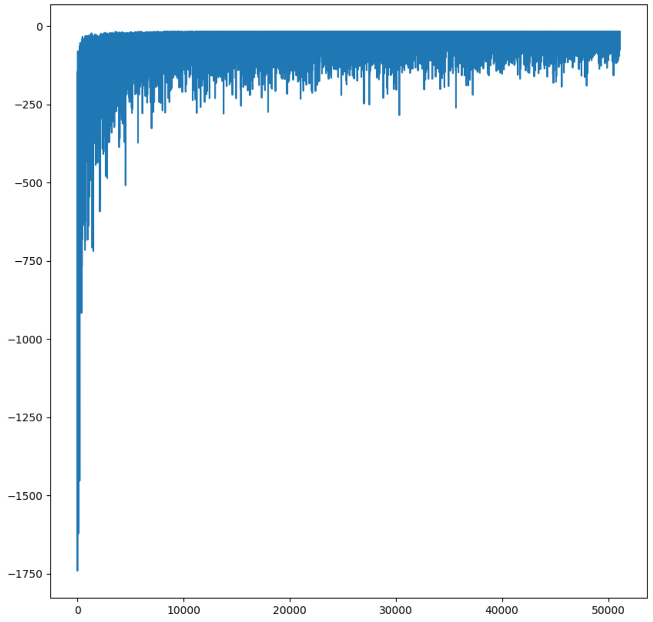
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    until S is terminal

- Simple model-free, TD control method.
- Learns the state-value action.
- Have to manually set the exploration-exploitation parameter.
- Suffers when the dimension is too large.



Number of steps until done



Cumulative reward for all agents

Training and Results on Q-Learning

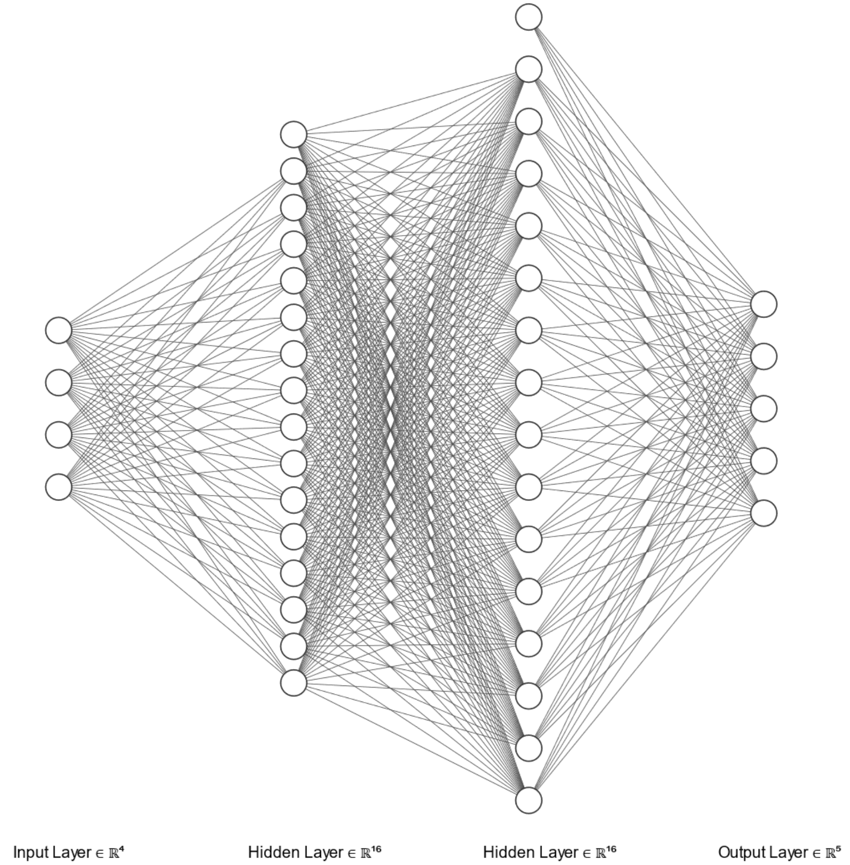
- Trained on 1 million steps, with epsilon rate of 0.1
- Each epoch is limited to max 500 steps

Deep Q-Learning

- Approximate the Q function using (usually) Neural Network.
- Needs a replay buffer to ensure independent, non-correlated training samples.
- Require a lot of fine-tuning parameters:
    - Exploration rate
    - Update frequency
    - Replay buffer size
    - Etc
- Model design is also complicated, and dependent on the task.
- Hardware requirement.
- To address the two last issues, we opt for a small environment, and a small model.

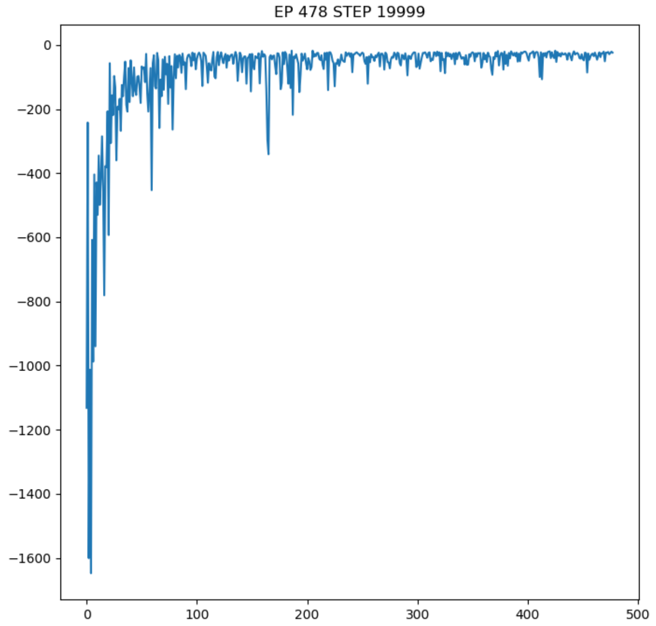
Model Architecture

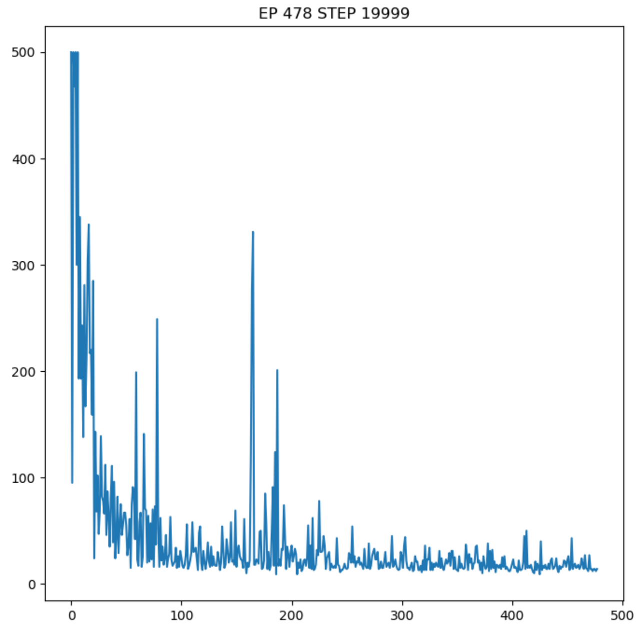
The FCN architecture (model not drawn to scale).



- Simple FCN with two layers, each of size 128.
- Using Rectified Linear Unit activation function.

Cumulative Reward





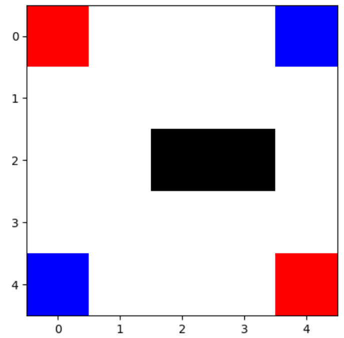
Step until solved

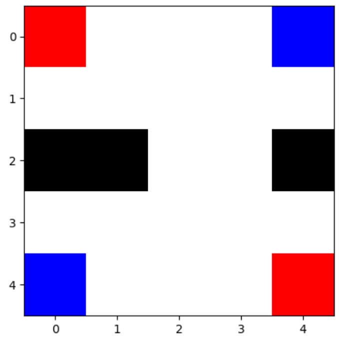
Training and Results

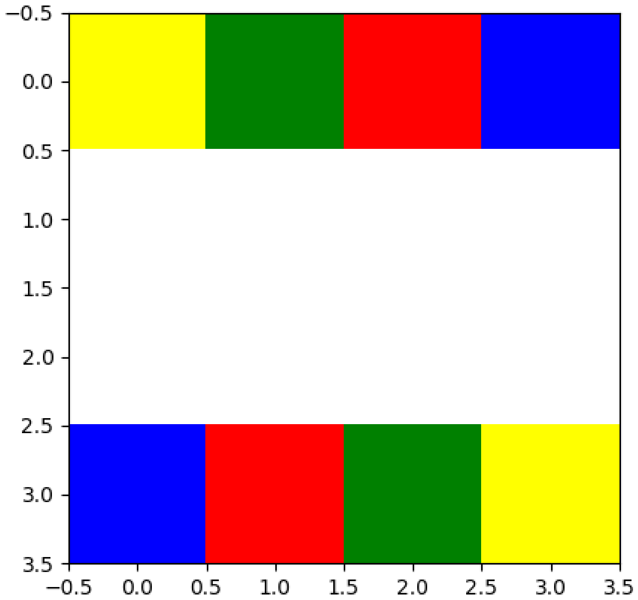
- Trained on 20000 steps.
- 5000 buffer size.
- Episode terminate at max 500 steps.
- Linear decay epsilon from 1 to 0.01 in 10000 steps.
- Update freezing weight every 1000 steps.

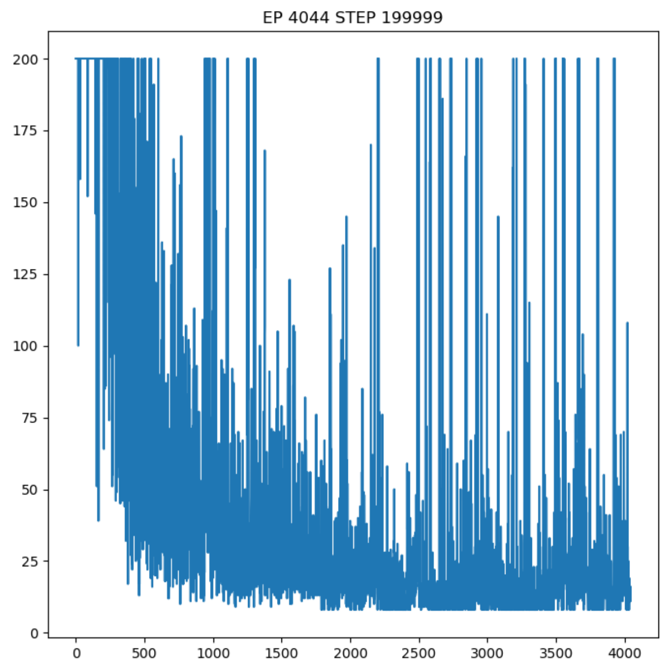
Adding flavor to the mix

- Can our simple model solve more complex tasks?
- Showing the varieties of the environment.
  - Maximum of 4 agents (when render is needed).
  - Adding randomized obstacle.

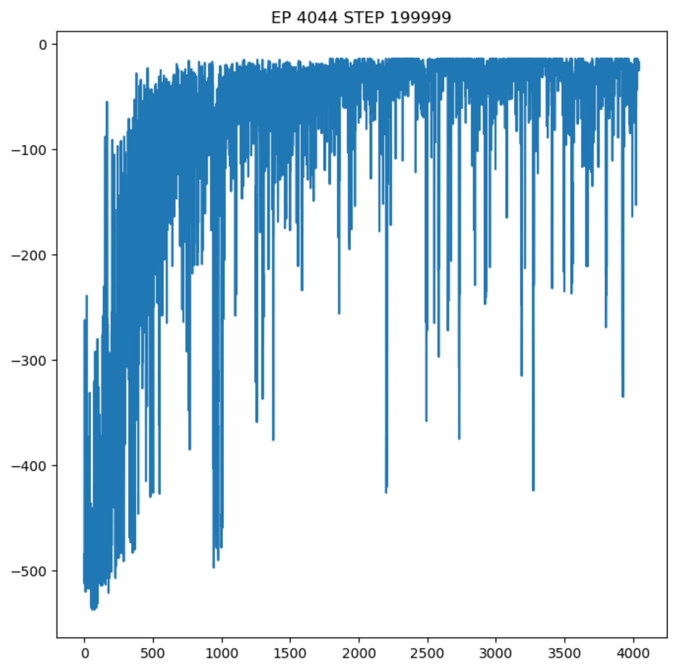




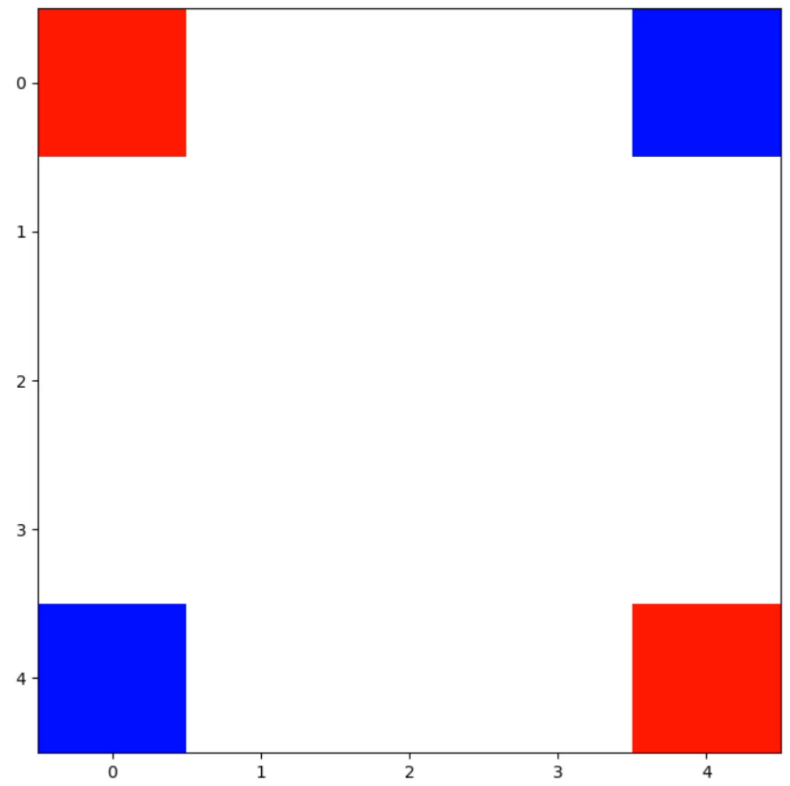




Steps to solve



Cumulative reward

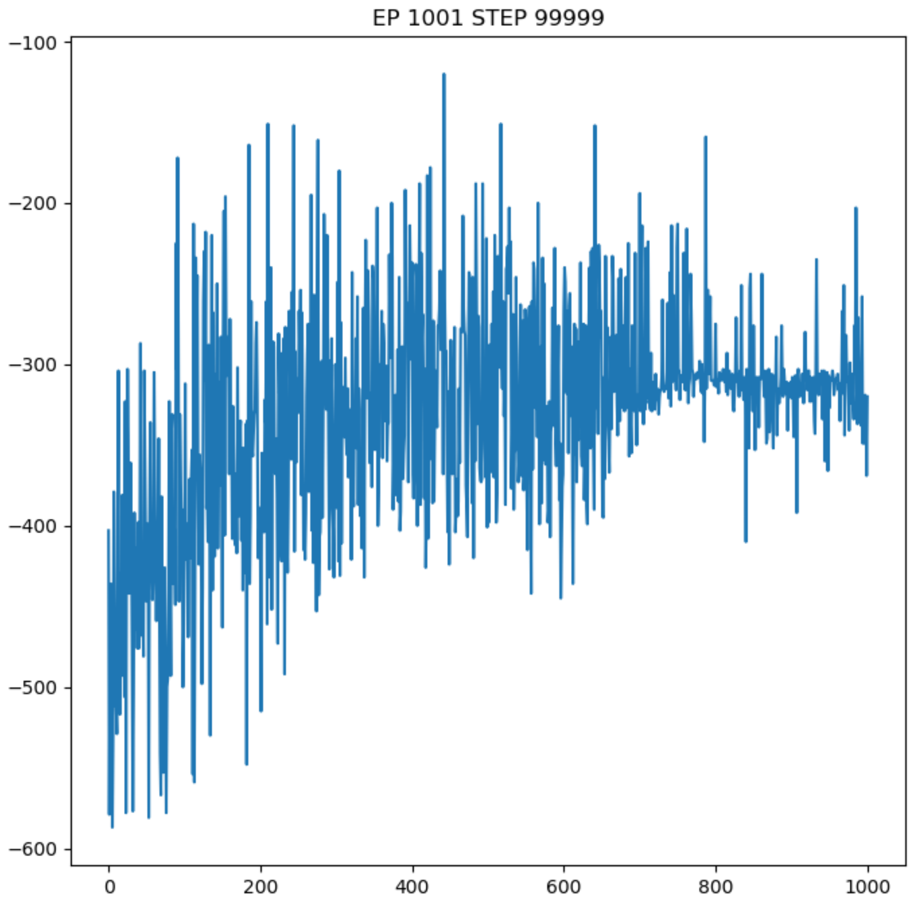


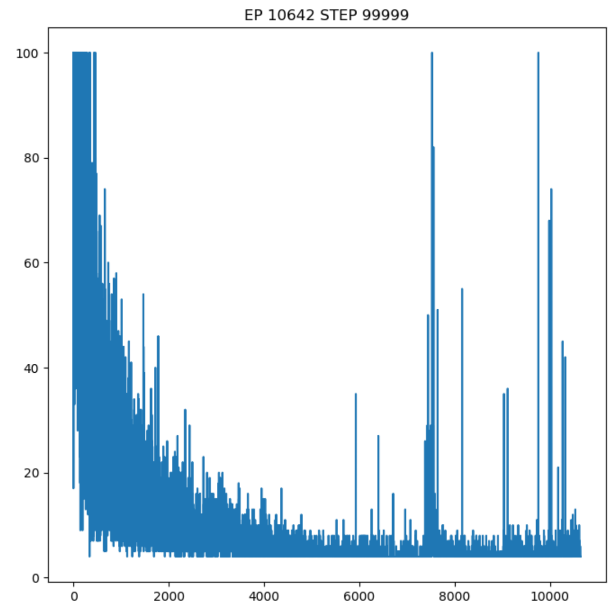
Navigate with randomized obstacle

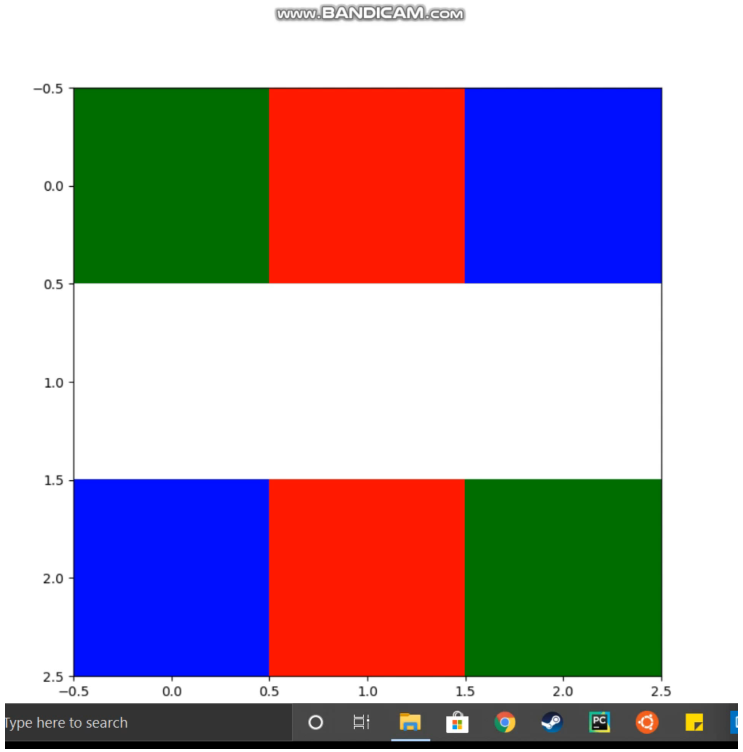
- 200000 training steps. 30000 replay buffer size
- Epsilon linear decay until 150000 steps.
- Change the obstacle every 10 episode (max 200 steps per episode).

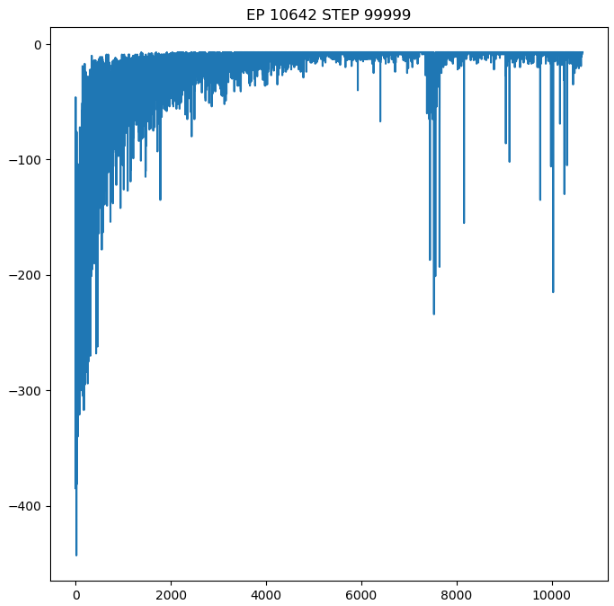
MORE AGENTS

- Agents seems to learn how to avoid collision, but not learned to reach target.
- Simply issue No-op to avoid collisions.





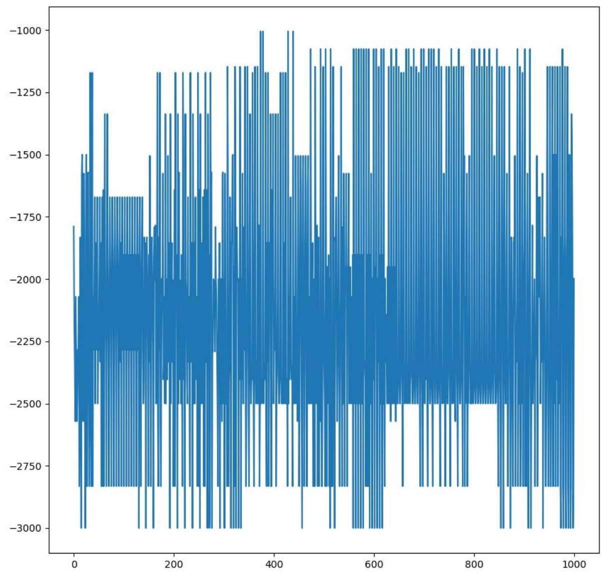


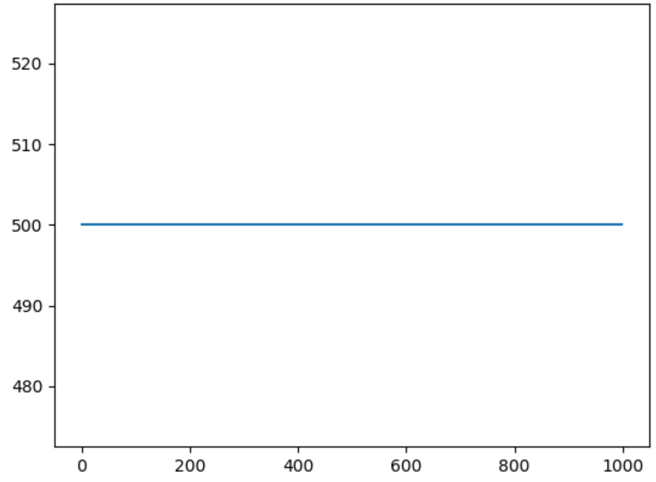


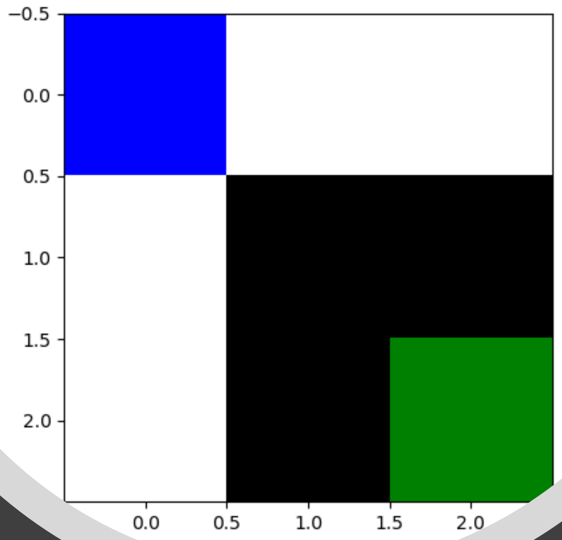
An easier version

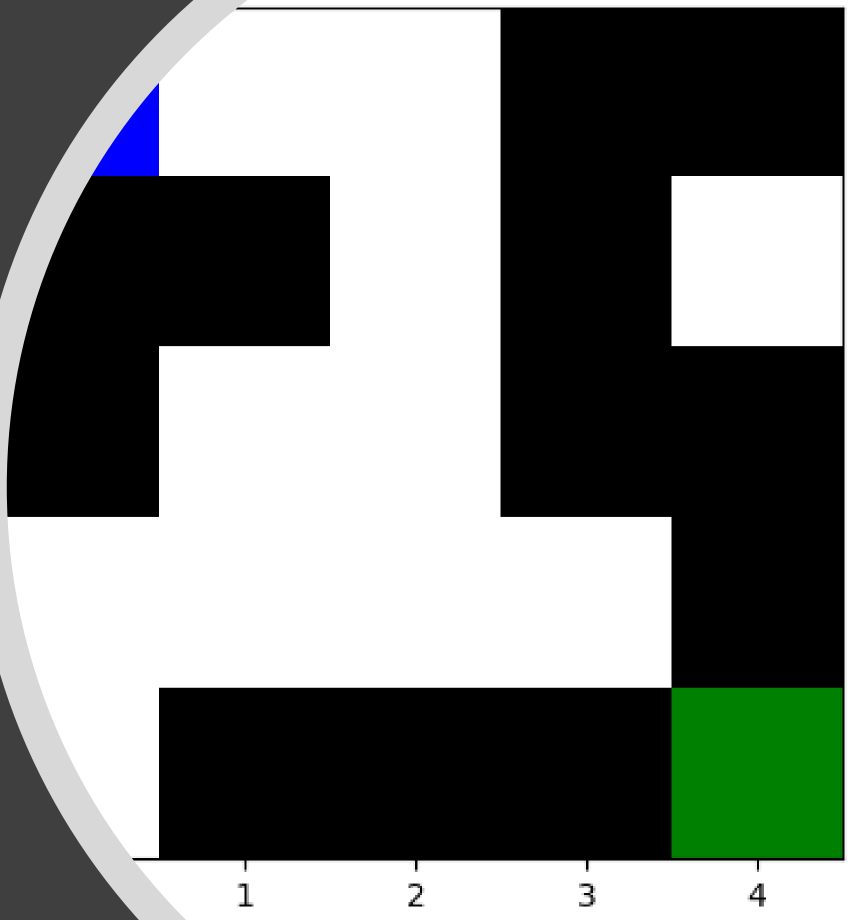
Some failed attempt using PG

- Not converge well due to the stochasticity of the policy, and the agents.









THANKS FOR YOUR ATTENTION

Project environment were revised after failed attempt of solving.