Mission Impossible: Deep Q Network Agent in DOOM

Yirong Tang, Shucheng Tian

1 Project Overview

For this project, we are going to implement a reinforcement learning (RL) agent with Deep Q Network and train the agent to play the classic 1993 First-Person Shooting game: "DOOM".

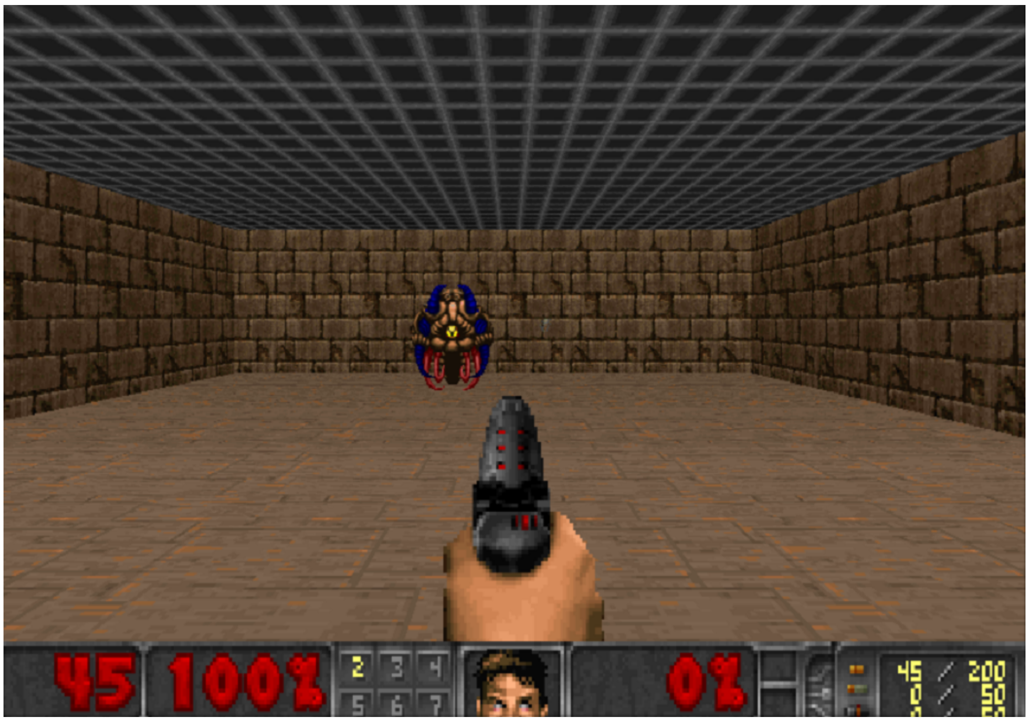


Figure 1: A Snapshot from the DOOM Game

To make the project manageable, we will not include the entire DOOM game; rather, we adopt a simple version where the environment is contained within a rectangular room, in which our agent and the monster take opposite sides of the room. The action space of the agent is limited to three actions: move left, move right, and shoot.

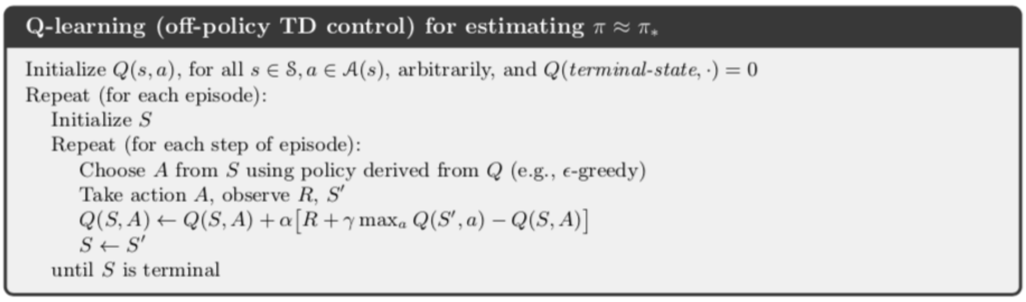
For each episode, a monster is spawned randomly somewhere along the opposite wall. The agent will repeatedly choose among the three available actions until either the monster is killed, or the episode reaches timeout.

Our main goal for this project is to successfully train the agent to identify the enemy, kill the enemy before time is up and as efficient as possible. Our secondary goal is to demonstrate the idea of Experience Replay, which is a technique often used with Deep Q Network to avoid forgetting previous experiences and reduce correlations between experiences.

After we have implemented the gaming agent, we intend to evaluate its performance based on two criteria: (1) the total reward the agent collects in an episode or game averaged over a number of games.[1], and (2) the policyâs estimated action-value function Q. To validate our evaluation, we will also be looking at three empirical evidence: (1) the percentage of the episode in which our agent successfully shoots the monster, (2) number of total shots fired to the number of monsters killed, and (3) the average time from the monster spawned to killed by our agent.

2 Background and Related Work

Q-Learning is an off-policy TD control algorithm and one of the early breakthroughs in reinforcement learning. The goal of Q-Learning is to learn a policy, which tells an agent what action to take under what circumstances. For any finite Markov decision process (FMDP), Q-learning finds a policy that is optimal in the sense that it maximizes the expected value of the total reward over all successive steps, starting from the current state. It has advantages such as that it does not require a model of the environment and can handle problems with stochastic transitions and rewards, without requiring adaptations.[4]



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

```
Initialize $Q(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize $S$
  Repeat (for each step of episode):
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\epsilon$-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
```

Figure 2: Q-learning Algorithm

However, as the size of state space becomes larger, producing and updating Q-table can be very ineffective. Therefore, we introduce a new idea called **Deep Q-Learning (DQN)**. Instead of using a Q-table, we implement a Neural Network that takes a state and approximates Q-values for each action based on that state.

The concept of Deep Q-Learning is purposed in a 2013 paper by DeepMind in a similar research project to use Deep Reinforcement Learning to play the Atari game. [2] In a 2015 Nature paper submitted by DeepMind, to derive efficient representations of the environment from high-dimensional sensory inputs, and use these to generalize past experience to new situations, researchers at DeepMind use recent advances in training deep neural networks to develop a novel artificial agent, termed a deep Q-network, that can learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning.[3]

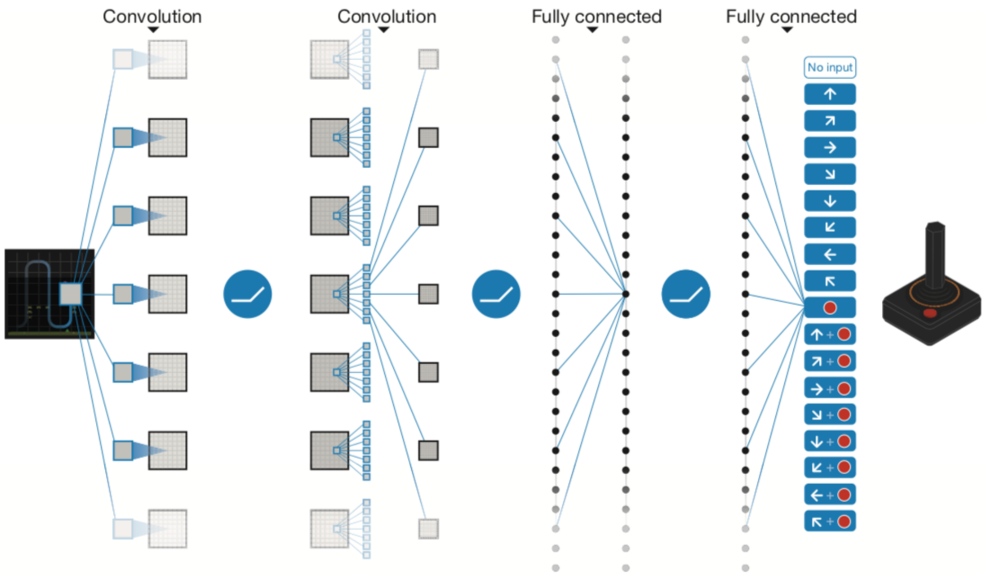


Figure 3: Illustration of A Convolutional Neural Network Approximating Q-Value of Given States

A similar research article is the paper published by DeepMind in 2013 that uses Deep Reinforcement Learning to play the Atari game - *Playing atari with deep reinforcement learning.*

3 Problem Formulation and Technical Approach

As mentioned in the previous section, our model is restricted to a simple version of the game DOOM. The action space is composed of three distinct actions: move left, move right, and shoot. The reward is given as follows:

- When the monster is killed, the agent receives +101 points
- Each time the agent pulls the trigger but misses the monster, the agent receives -5 points
- At the end of each episode, regardless of how it ended (either with the monster dead or the episode reached timeout, the agent receives -1 points.

The reward is designed with two specific goals: The reward of killing the monster is significantly greater than the penalty given to the agent, so it helps the agent learn the optimal policy of playing this game; The penalty of -1 is given at the end of each episode is to tell the agent not to choose the policy with which no shooting ever happens, just to avoid being penalized for missing the target. Once we have a working version of the Deep Q Network agent, additional rewards can be introduced. For example, for every second the monster is alive, the agent receives -1 point, and the agent would learn to minimize the killing time, resulting in a more effect agent.

The core of our agent's learning algorithm is a Deep Q Neural Network that takes a stack of four frames at each state as input, and passes the stacked frames (after some image processing) through the layers of the network and returns a vector of Q values for each of the three actions at that given state.

The equation we use for updating the Q value is the Bellman equation:

$$\underbrace{NewQ(s,a)}_{\text{New Q value for that state and that action}} = \underbrace{Q(s,a)}_{\text{Current Q value}} + \underbrace{\alpha}_{\text{Learning Rate}}[\underbrace{R(s,a)}_{\text{Reward for taking that action at that state}} + \underbrace{\gamma}_{\text{Discount rate}}\, \underbrace{max\, Q'(s',a')}_{\text{Maximum expected future reward given the new s' and all possible actions at that new state}} - Q(s,a)]$$

Figure 4: Equation for updating Q values

Since we are using Deep Q Network, we want to update the Neural Network's weights to reduce error.The error is calculated by taking the difference between our Q-target (maximum possible value from the next state) and Q-value (our current prediction of the Q-value). The equation is as follows:

$$\underbrace{\Delta w}_{\text{Change in weights}} = \underbrace{\alpha}_{\text{learning rate}}[\underbrace{(\underbrace{R + \gamma\, max_a\, \hat{Q}(s',a,w)}_{\text{Maximum possible Qvalue for the next\_state (= Q\_target)}}) - \underbrace{\hat{Q}(s,a,w)}_{\text{Current predicted Q-val}}}_{\text{TD Error}}]\ \underbrace{\nabla_w \hat{Q}(s,a,w)}_{\text{Gradient of our current predicted Q-value}}$$

Figure 5: Equation for updating Network Weights

After we set up our Deep Q Network, we will run 500 hundred episodes and update the model every 5 episode. The agent, once completed its training, will be evaluated for its performance. The evaluation metric will be discussed in the following section.

4 Evaluation and Expected Outcomes

In reinforcement learning, the evaluation is focus on the total reward the agent collects in an episode or game averaged over a number of games.[1] Based on our evaluation metric, we periodically compute total reward of each episode during training. Another, more stable, metric is the policy's estimated action-value function Q, which provides an estimate of how much discounted reward the agent can obtain by following its policy from any given state. We collect a fixed set of states by running a random policy before training starts and track the average of the maximum predicted Q for these states.

We will also try to evaluate our algorithm by comparing with the best performing methods from RL literature, such as **Sarsa** and **Contingency**. Since the game Doom has too many state, we are not sure about whether we have enough computational power to run traditional tabular methods. But comparing our method to traditional reinforcement agent is still on our plan.

Besides reinforcement agents, we also plan to compare the results gained by our DQL agent with human agent, which is measured by the median reward achieved after around two hours of playing the game Doom.

We expect that our DQN agent would outperform both traditional reinforcement agents as well as human agents in the game, based both on empirical finding and total reward gained per episode.

å·®ä¸åä°

References

[1] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[4] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.