Learning Montezuma’s Revenge from a Single Demonstration

Tim Salimans, Richard Chen (OpenAI)

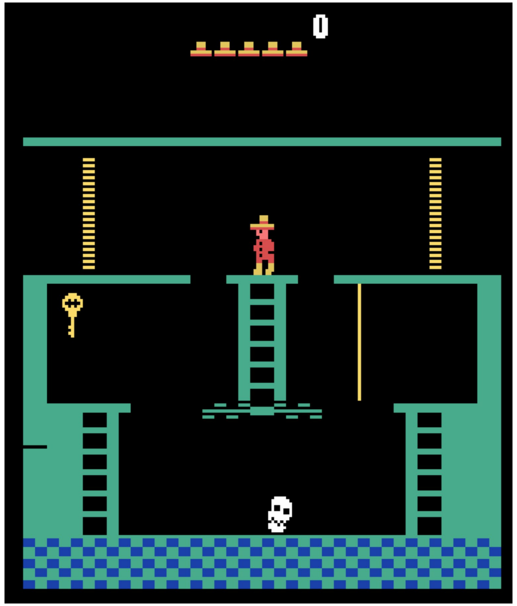
Deep RL Workshop, Neural Information Processing Systems (NeurIPS), 2018

Introduction

- Sparse reward is a common challenge in reinforcement learning.
- This occurs when an agent is in an environment where reward signals are infrequent.
- Sparse rewards can make it difficult for the agent to learn the optimal behavior.

Montezuma's Revenge

- Rules
  - Platformer game where the player controls an explorer.
  - The goal is to get as much score as possible before he loses all his lifes.
  - The player receives score when
    - picking up an item
    - killing an enemy using a sword
    - opening a gate using a key



Montezuma's Revenge

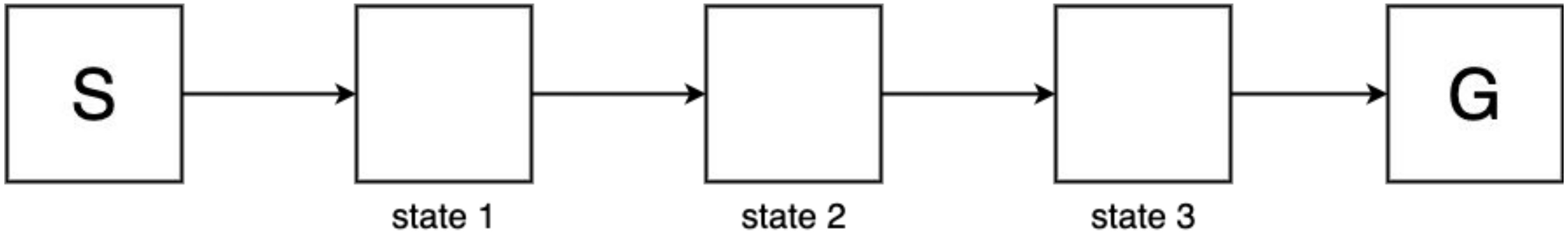
- One of the hardest games in Atari 2600
  - Scores of DQN agents

DQN	DDQN	Prior. DDQN	Duel.DDQN	Distrib. DQN	Noisy DQN
0	0	0	0	367	0

Matteo, H. et. al, “Rainbow: Combining Improvements in Deep Reinforcement Learning”, *arXiv preprint arXiv:1710.02298*, 2017

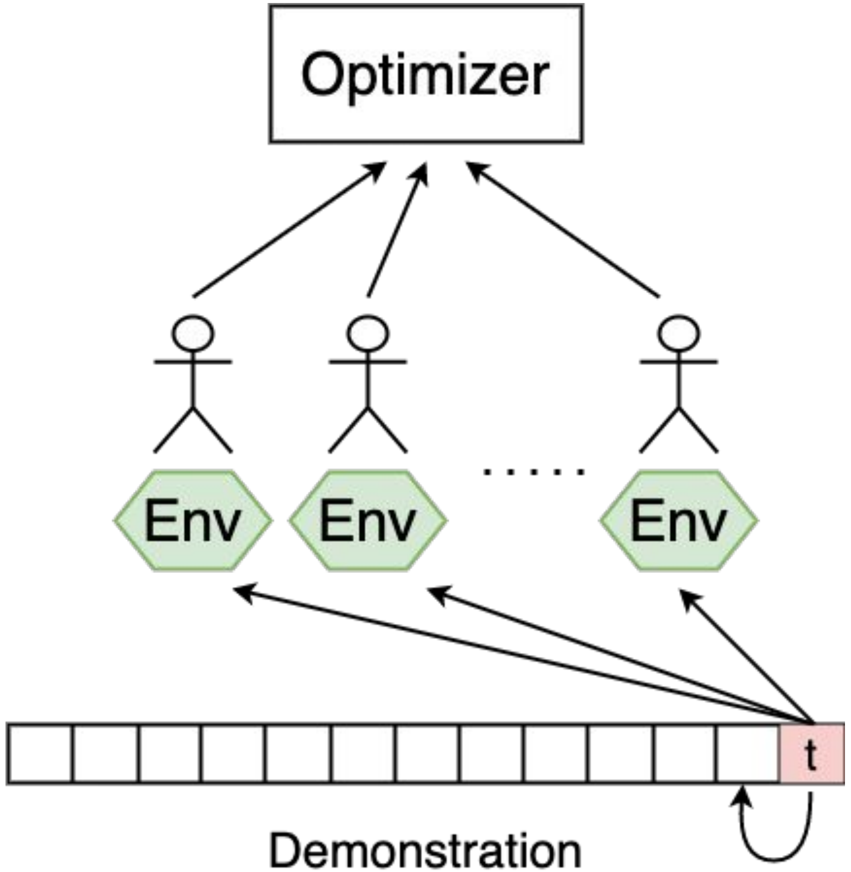
- In this game, if the agent performs an action randomly, it takes about 10^19 random action sequences to reach the first reward.

Method Idea



- Suppose you want to complete a game.
- You are given a demonstration of a successful completion of game play.
- You want to learn how to reach the goal by imitating the demonstration.
- First, you try to learn how to go from state 3 to the goal.
- Next, you try to learn how to go from state 2 to the goal. This is fairly easy because you already know how to go from state 3 to the goal
- Repeat this process until you reach the starting state. Now you've learned a whole path to complete a game.

Method



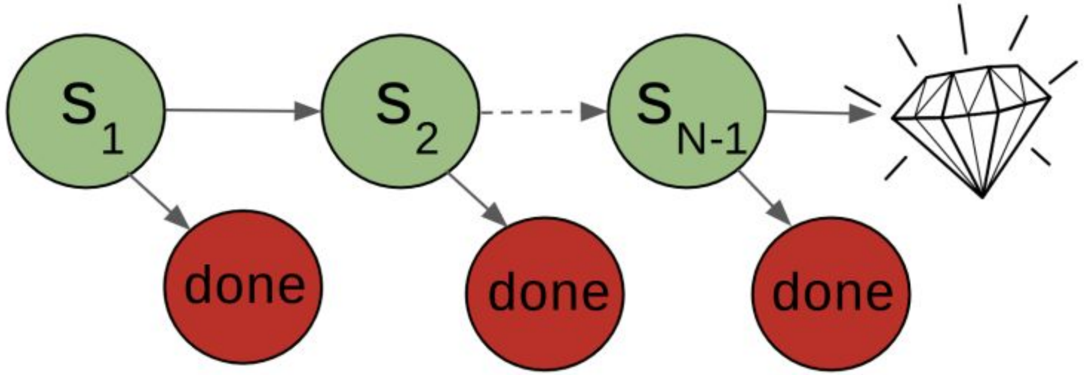
(3) Optimizer optimizes policy using trajectories sent from workers.

(2) Agent (worker) plays the game until it reaches the end of the game and then sends trajectory to optimizer.

(1) Initialize environments to state t in demonstration.

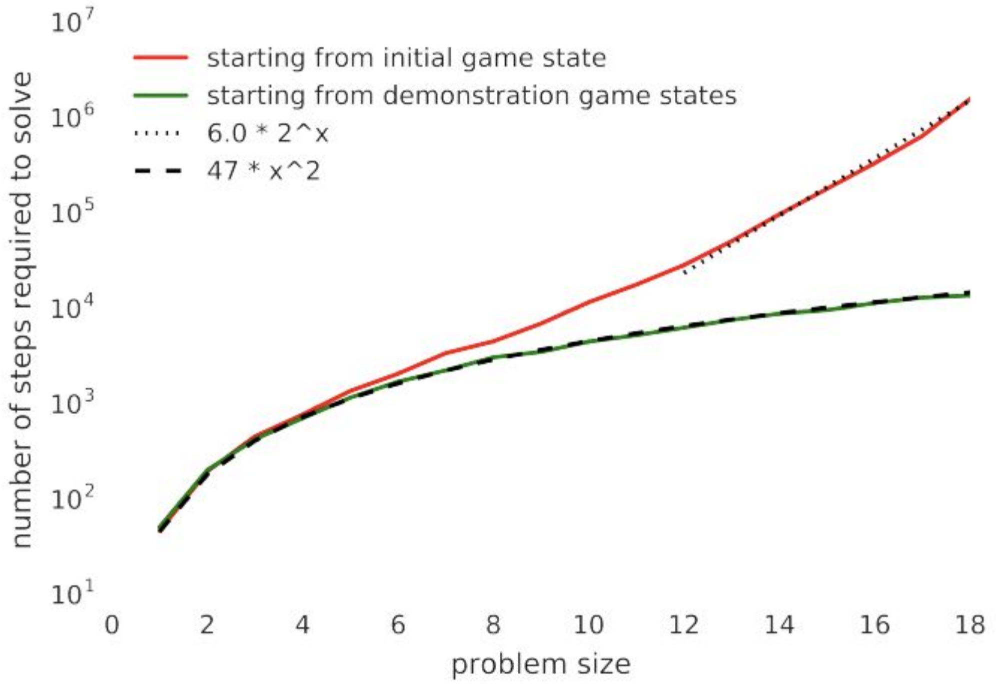
(4) Move t one step backward if the agents already played well. Go back to (1)

Efficiency of The Method



- Blind cliff walk
  - RL toy problem
  - The agent starts in state 1.
  - The agent has two actions.
  - One of these actions will take it to the next state.
  - The other action will make it fall off the cliff.
  - The agent receives reward only when it reached the diamond.

Efficiency of The Method



When starting from the initial game state, the agent requires O(2^N) steps to reach the diamond,

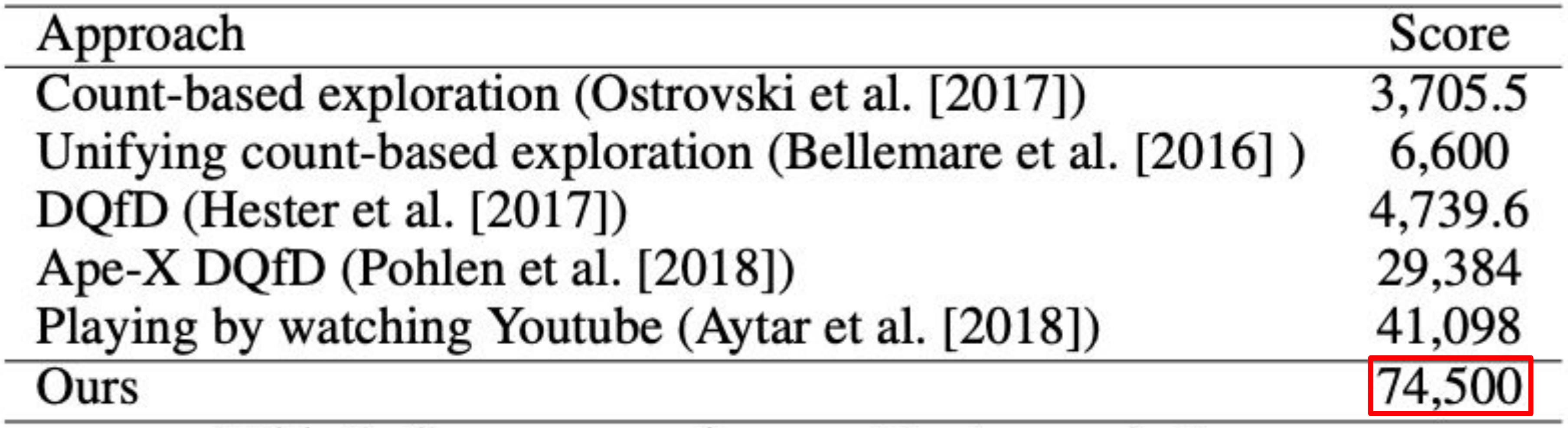
whereas if it begins from the demonstration states like their method, the agent only needs O(N^2) steps.

Experiment

- Implementation
  - Input is pixels
  - PPO optimizer
  - 1024 workers
  - 128 GPUs, 50 billion steps, took 2 weeks

- Demonstration
  - 15 minutes of game play
  - score was 71500

Result



Approach	Score
Count-based exploration (Ostrovski et al. [2017])	3,705.5
Unifying count-based exploration (Bellemare et al. [2016])	6,600
DQfD (Hester et al. [2017])	4,739.6
Ape-X DQfD (Pohlen et al. [2018])	29,384
Playing by watching Youtube (Aytar et al. [2018])	41,098
Ours	74,500

Table 1: Score comparison on Montezuma's Revenge

Remaining Challenges

- It is generally unable to follow the exact state and action from demonstration due to randomness. So, the agent needs to generalize between similar state.

  This works well for Montezuma's Revenge, but less well for other games in Atari that they tried.

  One reason for this may be that they require solving more complicated visual problems. The authors noticed some improvement when using larger and deeper neural network policies.