Playing Snake using Deep Reinforcement Learning

Sahil Johari

CSE 8340, Southern Methodist University

Abstract

Deep reinforcement learning (DRL) is poised to revolutionize the field of artificial intelligence (AI) and represents a step toward building autonomous systems with a higher level of understanding of the visual world. Currently, deep learning is enabling reinforcement learning (RL) to scale to problems that were previously intractable, such as learning to play video games directly from pixels.

Background

Reinforcement learning [2], explained simply, is a computational approach where an agent interacts with an environment by taking actions in which it tries to maximize an accumulated reward.

An agent in a current state (S_t) takes an action (A_t) to which the environment reacts and responds, returning a new state(S_{t+1}) and reward (R_{t+1}) to the agent. Given the updated state and reward, the agent chooses the next action, and the loop repeats until an environment is solved or terminated.

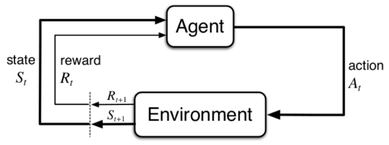

Fig 1: Principle of Reinforcement learning

Q-Learning is a technique that evaluates which action to take based on an **action-value** function that determines the value of being in a certain state and taking a certain action at that state [2].

$$Q_{t+1}(s_t, a_t) = \underbrace{Q_t(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \times \overbrace{\Big[\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1}, a_t)}_{\text{estimate of optimal future value}}\Big]}^{\text{learned value}} - \underbrace{Q_t(s_t, a_t)}_{\text{old value}}$$

Fig 2: Bellman Equation

Problem Statement

- Given a 2-dimensional environment space of size MxN, using a convolutional neural network (CNN), build a **Deep Q-Network** (DQN) [3] which takes 2 consecutive game frames as input (2, M, N) and output an action for the game.
- Solve the problem to maximize the game score.

Approach

Creating an Environment

The first step is to create an environment for our agent to operate on. This has been implemented in a simple way using Python co-routines [1] and will be rendered (or displayed) using *Pyplot* (from Matplotlib). The environment consists of:

- A *snake*, which is a 3x1 white block when initialized,
- A *food*, which is a 1x1 gray block,
- Four *boundaries*, which are 10x1 white blocks around the play area.

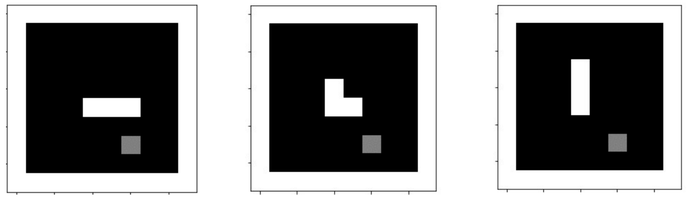

Fig 3: Snake environment

Modeling an Agent

- An agent would be a model which consists of a mechanism to play the game. In this case, the agent would figure out a way to learn and play Snake.
- The agent uses a *Convolutional Neural Network* to build a DQN.

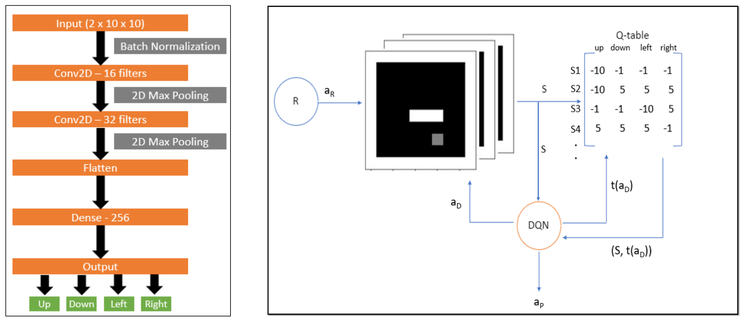

Fig 4: Training Pipeline – CNN model for DQN (left); Training architecture (right)

- Training Pipeline
  - Given a_D actions along with some random actions a_R to generate a batch of states S.
  - DQN takes states S as input to predict an action and reward $t(a_D)$, and creates a Q-table.
  - The Q-table consists of state-action mapping in terms of rewards, which is fed to the DQN as (S, $t(a_D)$).
  - DQN predicts the output action a_P.

Results

Scores per gameplay

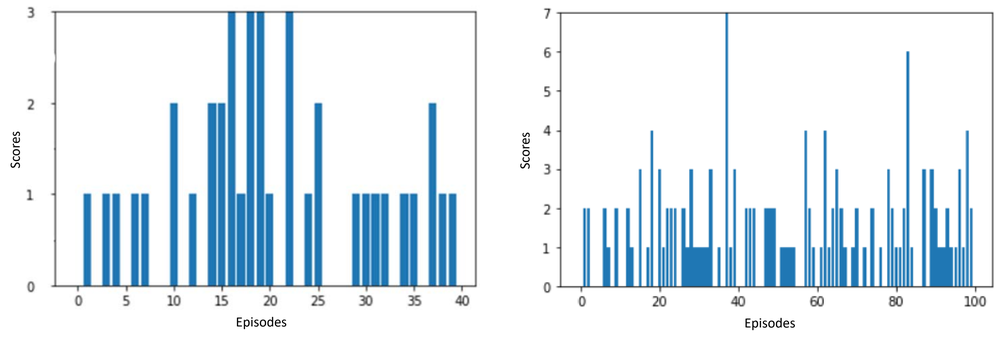

Fig 5: Maximum score of 3 achieved after training for 5000 iterations and tested for 40 episodes (gameplays)

Fig 6: Maximum score of 7 achieved after training for 10000 iterations and tested for 100 episodes (gameplays)

Ideally, the Snake could make a score of 61, given the environment dimensions of 10x10 and a set of 4 walls around it with 1 unit thickness. However, the snake can have a maximum length of 8 in either horizontal or vertical orientation. Achieving **a score of 7** is a good result so far!

Conclusion

- Applied the concept of deep reinforcement learning on the classic game - Snake.
- Used an approach called Q-learning, which is based on value functions that estimates the expected return of being in a given state.
- Extended this approach to deep Q-network and used a convolutional neural network to implement it. Using this approach, a maximum score of 7 was achieved.

Future Work

- This project can be further extended using concepts like Policy search and Actor-Critic method [2].
- An interesting implementation would be to incorporate Genetic algorithm to create a population of agents and filter out the best through various generations.

References

[1] Basic Reinforcement Learning by Víctor Mayoral Vilches, https://github.com/vmayoral/basic_reinforcement_learning/blob/master/tutorial6/examples/Snake/snake.py

[2] Introduction to reinforcement learning and OpenAI Gym, https://www.oreilly.com/learning/introduction-to-reinforcement-learning-and-openai-gym

[3] Deep reinforcement learning: where to start, https://medium.freecodecamp.org/deep-reinforcement-learning-where-to-start-291fb0058c01