Crossy Road Reinforcement Learning

Neel Shah, Anish Bagri, Nathan Sankar, Akul Gokaram, Aditya Sharda

May 7, 2024

1 Introduction

Crossy Road is an arcade video game built around the age-old joke "Why did the chicken cross the road?" In the game, the chicken (controlled by the player) has to cross the road without getting hit by vehicles.

We want to develop a reinforcement learning (RL) game agent capable of playing Crossy Road. The game's endless and random nature makes it a great candidate for RL.

The agent will learn to maximize its score by getting the chicken to cross the road and avoid obstacles in its path, with the ultimate goal of crossing the road as many times as possible without collisions. Once the agent is capable of successfully getting the chicken to cross the road and reach the goal position, another goal could be to minimize the time it takes for the chicken to cross the road.

2 Configuration

We'll start by importing the Python libraries necessary for this project and configuring some things.

[13]:
```
import gymnasium as gym
```

3 Crossy Road Environment

Our first step is to implement a Crossy Road environment, which will encapsulate our representation of the reinforcement learning problem that the game poses.

For this, we will utilize the Gymnasium library (a fork of the OpenAI Gym library), which provides a standard API for RL and various reference environments.

Specifically, we will use the Freeway environment, which models an Atari game that closely resembles Crossy Road. This gives us a Pythonic interface to work with, which we can later use to develop RL models and create an agent that can play Crossy Road successfully.

3.1 Initializing the Environment

We will start off by initializing the environment using Gymnasium.

We pass in the following arguments (documented here) to specify the environment:

Environment Flavor:

The environment `id`, `mode`, and `difficulty` combine to specify the specific flavor of the environment:

- `id="ALE/Freeway-v5"`: simulates the Atari game Freeway via the Arcade Learning Environment (ALE) through the Stella emulator
- `mode=0`: selects Game 1 (Lake Shore Drive, Chicago, 3 A.M.) as the map to use
- `difficulty=0`: selects the default difficulty setting

Stochasticity:

As stated in the documentation:

> As the Atari games are entirely deterministic, agents can achieve state-of-the-art performance by simply memorizing an optimal sequence of actions while completely ignoring observations from the environment.

To combat this, we use `frameskip` and `repeat_action_probability`:

- `frameskip=4`: enables frame skipping (sets the number of frames to skip on each skip to 4)
- `repeat_action_probability=0.25`: enables sticky actions (sets the probability of repeating the previous action instead of executing the current action to 25%)

Simulation:

The parameters `full_action_space` and `render_mode` are used to specify how the environment is simulated:

- `full_action_space=False`: limits the action space to the 3 legal actions we will actually use instead of all 18 possible actions that can be performed on an Atari 2600 console
- `render_mode="human"`: specifies that the game should be rendered in human mode, displaying the screen and enabling game sounds

[14]:
```
env = gym.make(
    id="ALE/Freeway-v5",
    mode=0,
    difficulty=0,
    obs_type="rgb",
    frameskip=4,
    repeat_action_probability=0.25,
    full_action_space=False,
    render_mode="human",
)
```

Now, we are ready to learn a little more about how out environment is implemented.

3.2 Observations

Let's start with the observation space.

[15]:
```
env.observation_space
```

[15]:
```
Box(0, 255, (210, 160, 3), uint8)
```

This observation space represents the RGB image that is displayed to a human player.

3.3 Actions

Next, let's move on to the action space.

[16]:
```
env.action_space
```

[16]:
```
Discrete(3)
```

This action space represents the actions that the chicken can take in each step:

Value	Meaning
0	NOOP
1	UP
2	DOWN

3.4 Rewards

Finally, let's move on to the reward range.

[17]:
```
env.reward_range
```

[17]:
```
(-inf, inf)
```

We can see that the reward range is $(-\infty, \infty)$.

We don't really know the specifics, but more information can be found in the game manual.

4 Reinforcement Learning

5 Conclusion