The PlayStation Reinforcement Learning Environment (PSXLE)

Iñigo Munárriz, Aitor González, Carlos Domínguez

September 2020

Index

Index

Abstract

- New way for evaluating **Reinforcement Learning algorithms** through games.
- Using a modified version of the **PlayStation 1 emulator** connected through a **Python API**.
- **Playstation Learning Environment (PSXLE)** supports the **OpenAI Gym interface**.

Index

- Abstract
- Introduction
- Playstation and Kula World
- Implementation
- Rewards
- Evaluation
- Conclusions

Introduction

- **Reinforcement Learning (RL)** is a form of **Machine Learning** which uses a system of **rewards** and evaluates how **agents** interact with a given **environment**.
- **Agents** interact with the **environment** through actions that are performed based on the **state encoding**.
- **The state encoding** is the way agents **"perceive" the environment** in a given instant, getting only the information they need.

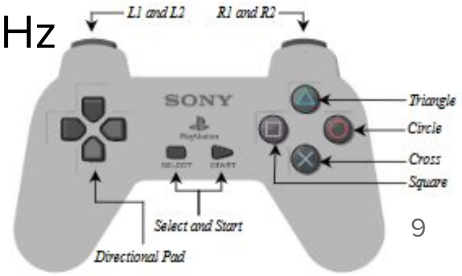
Introduction

- **Deep Q-Networks (DQN)** are usually used so **agents** can interpret even **complex states** of the environment.
- **Use of computer games** provide many advantages such as the simplicity of the environments and the "**score**" many games have.
- The **goal** is to prove PlayStation environment to be interesting in **RL**.

Index

VS

Atari-2600 (1977)		Playstation 1 (1994)
128 B	**RAM**	2 MB
128	**Displayable colours**	16.6 million
1.19 MHz	**CPU**	33.9 MHz
5	**Controller buttons**	14



Kula World (1998)

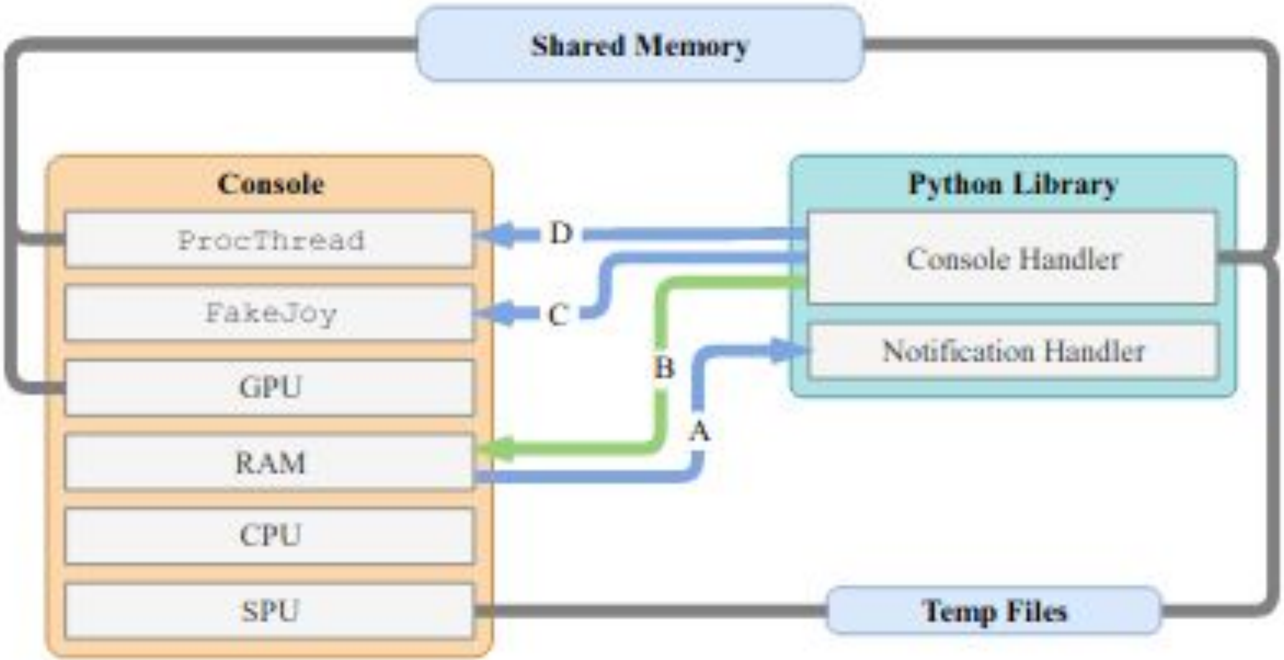
1. Control a ball
2. Collect:
    - Coins
    - Fruits
    - Keys
3. Arrive exit platform

Index

Implementation

- **PSXLE** is a toolkit for training agents to play Sony PlayStation 1 games.
- **PSXLE** is designed following the standards set by ALE (Arcade Learning Environment).
- Uses an **open source PlayStation emulator** with the necessary **modifications**.
  - Adding **Inter-Process Communication (IPC)** tools.
  - Adding a **Python based console API**.
    - Translates game actions into console functions.
    - Console functions can then be used by OpenAI Gym.

Implementation



A visualisation of the Inter-Process Communication used in PSXLE.

Implementation

The **Console API** supports four primary forms of interaction:

1. **General:**
    a. **run** and **kill** (the emulation process).
    b. **freeze** and **unfreeze** (the emulator execution).
    c. **speed**: sets the speed of the emulation relatively to the default speed of it.
2. **Controller:**
    a. **hold_button** and **release_button**: simulates the press down and the release of a given button.
    b. **touch_button**: simulates pressing a button (holding for a little amount of time and releasing).
    c. **delay_button**: adds delay between successive control events.

Implementation

The **Console API** supports four primary forms of interaction:

3. **RAM:**
    a. **read_bytes** and **write_byte**: read and write to console memory.
    b. **add_memory_listener** and **clear_memory_listeners**.
    c. **sleep_memory_listener** and **wake_memory_listener**
4. **Audio/Visual:**
    a. **start_recording_audio** and **stop_recording_audio**: controls when the console records audio.
    b. **get_screen**: returns an array with the visual output of the console.

Implementation

- **OpenAI Gym** uses three game abstraction methods: **reset, step** and **render**.
- However, **step** returns a tuple (the action performed in the game state, the rewards, etc) and all the elements of the tuple must be returned at the same time.
- Usually, **frame skipping** is the answer, when skipping we need less **state encodings.**
- Different moves can last different amount of times, so an **asynchronous** approach is needed.
- Uses an extra variable to indicate when the move is over.

Implementation

Four main actions a player can do in **Kula World**:

1. **Move forward.**
2. **Look right.**
3. **Look left.**
4. **Jump forward.**

However, this moves are not abstracted into **state encoding** literally, instead, some data of them is gathered after each move is performed.

Implementation

Contents of the **state encoding** after each move:

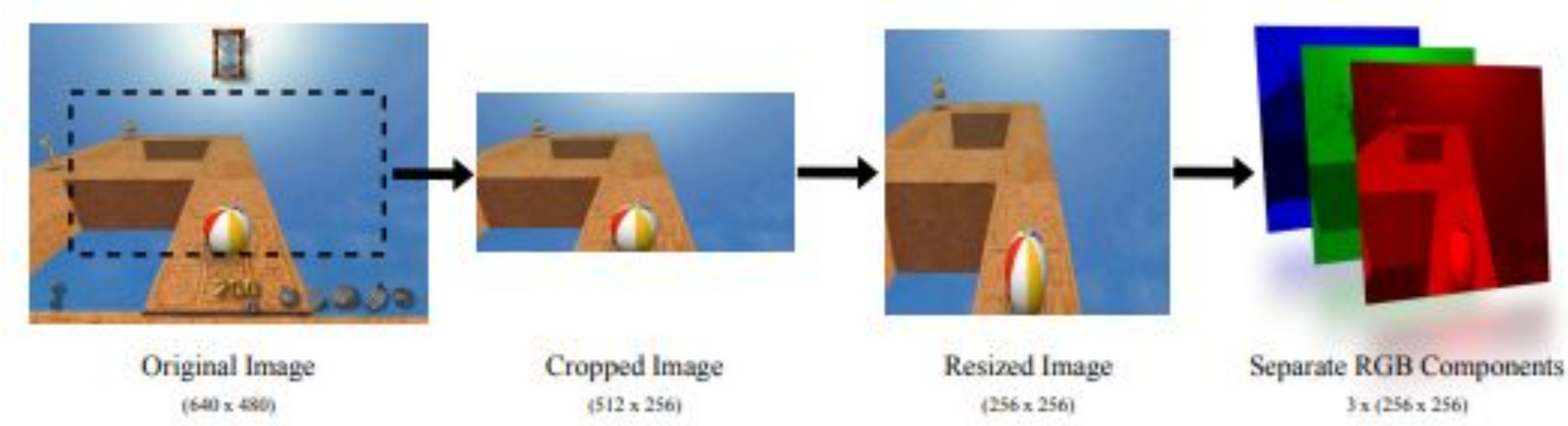
- **RGB** array.
- The **reward**.
- **playing**: which indicates if the player is still ‘alive’.
- **clock**: remaining seconds to complete the level.
- **sound**: and array which describes the audio output (or None if no audio).

Implementation

Contents of the state encoding after each move:

- **duration_real**: time the move took to be done.
- **duration_game**: remaining time that the move took to complete, relative to the in-game clock.
- **score**: the score achieved so far in a current level.

Implementation



How the RGB array is processed.

Index

- Abstract
- Introduction
- Playstation and Kula World
- Implementation
- Rewards
- Evaluation
- Conclusions

Rewards

- **RL** needs the **reward system** or **function**.
- Reward triggers the **optimal behaviour** of an **agent**.
- The higher the **reward** the more **desirable** the approach the **agent** selected is.
- Any event that leads to a **bad performance** is **punished**.
- At the end, **reward** can be considered as an **optimization value**.
- Using **Kula World**, a **function** is designed to transform the game **score** into a numerical **reward**.

Rewards

Reward function		
Event	Score change	Reward
Coin collect	+250	0.2
Key collect	+1000	0.4
Fruit collect	+2500	0.6
Win level	-	+1
Lose level	-	-1

Index

Evaluation

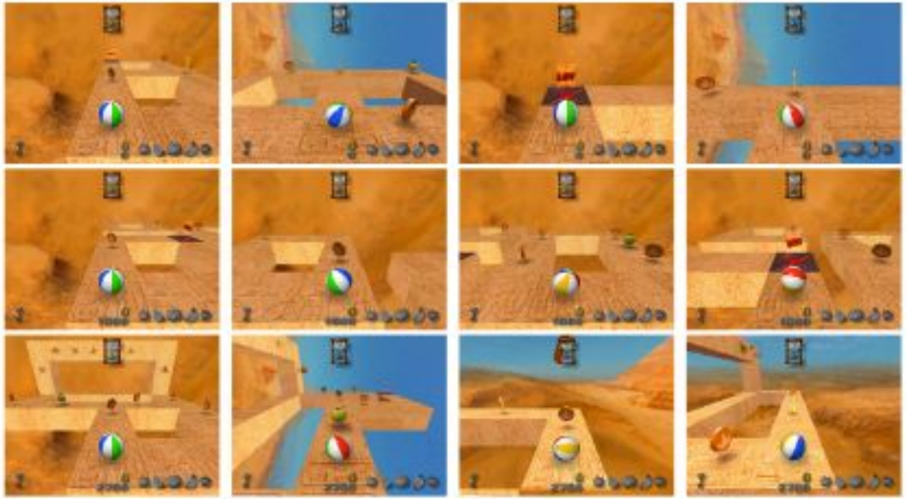
Kula World and **PSXLE** were used with **deepq** and **ppo2** from **OpenAI Baselines**

Problem?

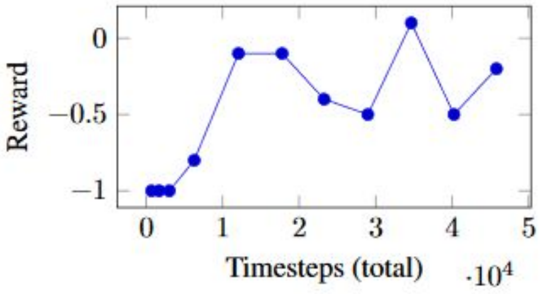
Agent starts at the same situation in each episode.
The agent simply **learns** how to beat that level in the shortest time with the highest score.

Solution?

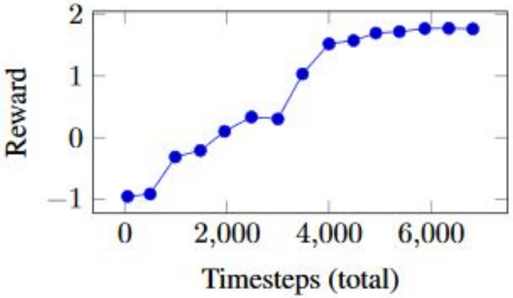
Create additional starting positions using the **save-state option**. (kula-random-v1)



Different random starting positions.



(a) The deepq baseline.

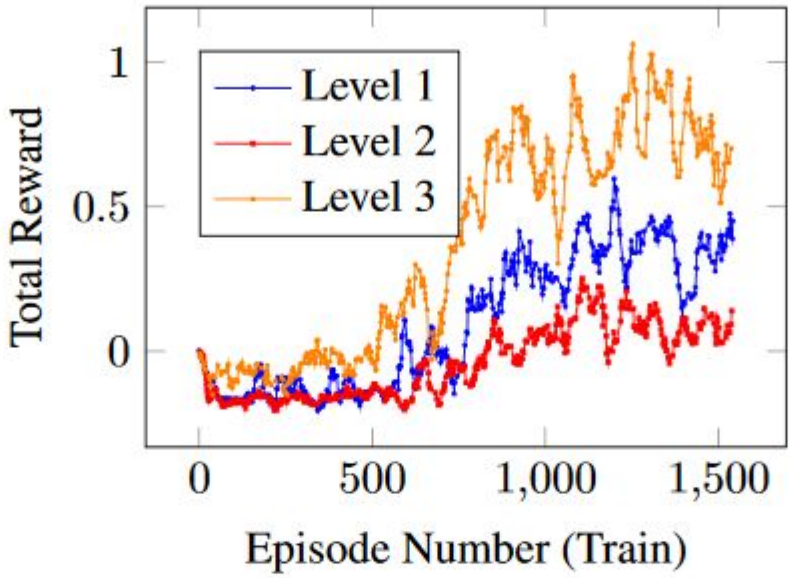


(b) The ppo2 baseline.

Evaluation

Interesting results

- **Level 2**: jump decision
  - Harder for IA
- **Level 3**: New physics
  - Harder for humans



Index

Conclusion

- **DeepQ** and **PPO2 RL algorithms** show that they perform vastly differently in **Kula World**.
- This approach can be used with **any other game**.
- By using **PSXLE** new **environments** can be **evaluated**, with more complex and **richer state spaces**.
- Games can become optimal environments in which RL algorithms effectiveness can be evaluated.

The End

Bibliography:
https://arxiv.org/pdf/1912.06101.pdf
carlospurves/psxle: A Python interface to the Sony PlayStation console.
Reinforcement learning - Wikipedia

Authors:
Iñigo Munárriz, Aitor González, Carlos Domínguez