**Project Notes**

Literature

Hierarchical reinforcement learning for self‐driving decision‐making without reliance on labelled driving data

* Independent Manoeuvre Selection and Motion Control
* APRL – Asynchronous Parallel RL – Each learner has own policy network, average is taken at each iteration, learners synchronise form shared network before making new decisions

Hands-On Reinforcement Learning with Python

* Reinforcement Learning Basics – Policy Function, Value Function, Agent-Environment Interface, Q Function, MDP, Bellman Equation, Monte Carlo Methods
* Policy Optimisation
* Feature exploitation vs Feature exploration

Deep Learning, Reinforcement Learning and World Models

* Generalised Policy Updates
* World Models

Lightweight 3D hand pose estimation by cascading CNNs with reinforcement learning

* CNN and feature extraction
* Feature exploitation vs Feature exploration

## Installing Mario-env

Was using python 3.12, needed Descartes package -> Pillow -> zlib.

Zlib not recognised by pip, tried .exe install, that didn’t work.

Found .whl archived package at <https://www.lfd.uci.edu/~gohlke/pythonlibs/#pillow>

Only compiled versions were python version <=3.11

Uninstalled python 3.12 and installed 3.11,

Installed pillow package on 3.11

## Changes to Mario Env

As I will be using a different approach so these trajectories are not as necessary

Removed all references to trajectories in state-handler

Removed Pangolin visualiser

## Key Memory Locations

|  |  |  |
| --- | --- | --- |
| Location | Offset | Description |
| 0x809BD70C | 0x61 | Moving Direction  (1: forward, 2: backward) |
| 0x809BD70C | 0x3C | Steering Direction  (0: left, 7: straight, 14: right) |
| 0x809BD730 | 0xF8 | Current Lap Completion  (float 0-3) |
| 0x809BD730 | 0x1B9 | Minutes |
| 0x809BD730 | 0x1BA | Seconds |
| 0x809BD730 | 0x1BC | Third-Seconds |
| 0x809C2EF8 | 0x40 + 0x0 | X Position (float) |
| 0x809C2EF8 | 0x40 + 0x4 | Y Position (float) |
| 0x809C2EF8 | 0x40 + 0x8 | Z Position (float) |

Source - <https://github.com/JackWBoynton/mariokart-rl/README.md>

Dolphin Config : GFX.ini -> Added `DumpFramesAsImages = True` in `[Settings]`

: Added hotkeys for frame advancing, learning can run independent of game framerate. Can be done not in real time

Installed Tesseract OCR Engine <https://github.com/tesseract-ocr/tesseract>

Funky Flame Runner: MT = 113 ,Wheelie = 97 ,Normal = 84

If I choose to advance the emulator frame by frame to allow more time for processing, then I will need to take into account the fact that dolphin only recognises inputs during a frame change. So the input would need to be held, frame advanced, then released. This approach could be very interesting to show the process of what happens each frame.

Charged MT = 270

Game version – ntsc-u

## Gecko Codes

To display relevant in-game information, a gecko code is used. This is a type of cheat code that is often used by people performing TASs (Tool-Assisted Speedruns) when they need to know the exact state of the game, which will also be very useful for me. These codes are supported by the emulator and can be activated/disactivated using the GUI.

The format for the Values I will be using is as follows

* XZ Velocity – A decimal number always to 2 DP, surprisingly this stays positive even when moving backwards
* Race% - A decimal number always to 5dp
  + Number before decimal is important too
* MT – A whole number between 0 and 270

To process the information displayed by the ‘TAS Debug Panel’ created by [Melg, stebler, Seeky, SwareJonge, kierio04, Gaberboo] I need to extract the text from each frame. To do this I used Google’s Tesseract Optical Character Recognition engine. Running the game at native (640x528) resolution gives uncertain results, so to improve the accuracy I increased the resolution of the game in the emulators settings. This would give the framedumps a higher resolution as the emulator dumps frames at the resolution it is running at. When deciding the crop regions, I made sure to leave a small border around the character. This along with the configuration settings helps tesseract to provide accurate results. These configurations include setting the psm (page segmentation mode) level which tells tesseract the high level format of the text within the image, such as paragraphs, single lines or single characters. For my use case this is set to a single line (level 6). Additionally I enforced a whitelist of the numerical characters. This is mainly to avoid confusion between similar looking pairs of characters such as 0 and O and I and 1. I decided to not include the decimal point since I can divide by the appropriate amount to get the numbers into the expected formats.

## Dolphin wrapper

* Init
  + Open dolphin
  + Assert config options – framedumps, input recording
  + Load default savestate
  + Pause emulation
* Step
  + Read frame
  + Calculate estimated reward for each input
  + Update Q function?
  + Return input with highest reward
* Reset
  + Load default savestate
  + Reset controller
* Terminate
  + Return sum of rewards
  + Stop emulation
  + Reset controller
  + Wipe framedumps folder

## Controller

* Reset
* Press button
  + Button name
* Angle stick
  + Stick
  + Anlge (0-14)

## Frame Reader

* Open file
* Greyscale
* Downsample
* Return information

