MarIQ Manual

# Introduction

This manual explains, roughly, how to install, train and run MarIQ, a deep Q learning implementation for playing Super Mario Kart.

My video about MarIQ: <https://www.youtube.com/watch?v=Tnu4O_xEmVk>

### Table of Contents

[Introduction](#_ahf2g96sw0ob)

[Installation](#_2nxpcnk175qr)

[Requirements](#_9t98a234w8me)

[Placing Files](#_2g2kz4q1u2r5)

[Emulator Setup](#_6pk7xzin7be4)

[Running the Neural Network](#_m456qyevx5t3)

[Running the Python Script](#_mhksc5atl3vl)

[Displaying Extra Information](#_zef0l6vjo78j)

[Further Options for Exploring MarIQ](#_tntlwou5st1x)

[Learn.py Command Line Options](#_9cktu6377o8h)

[Learn.py Config Files](#_ofmpjekx22jr)

[QPlay Lua Script](#_dusf5ln6hc3z)

# Installation

## Requirements

The following software is required to run MarIQ. I’ve included the version numbers that I have installed on my computer. Some of the requirements are stricter than others. See [here](https://www.tensorflow.org/install/) for more information on installing TensorFlow.

* [Python 3.5.0](https://www.python.org/downloads/release/python-350/) (64bit)
* [CUDA Development Tools 8.0.61](https://developer.nvidia.com/cuda-80-ga2-download-archive) (64bit)
* [CuNN 6](https://developer.nvidia.com/rdp/cudnn-download) (64bit)
  + For some reason downloading this requires creating an account.
  + With the release of Tensorflow 1.4, you may need CuNN 7 instead.
* [Visual C++ 2015 Redistributable](https://www.microsoft.com/en-us/download/details.aspx?id=48145)
* [tensorflow-gpu/tensorflow 1.3](https://www.tensorflow.org/install/install_windows) (can be installed with *pip* etc.)
* pygame (can be installed with *pip* etc: pip install pygame)
* [Bizhawk 1.12.2](https://github.com/TASVideos/BizHawk/releases/download/1.12.2/BizHawk-1.12.2.zip)
* Super Mario Kart ROM.
  + You’ll need to acquire dsp1b.rom as well, and install it into the Bizhawk.

## Placing Files

Download this zip file: <https://sethbling.s3-us-west-2.amazonaws.com/Downloads/MarIQ.zip>

The zip file contains two folders: **MarIQ** and **QLearning**.

* **MarIQ** contains the Lua scripts that plug into the Bizhawk emulator. In your Bizhawk installation, there’s a Lua folder. Place the MarIQ folder within that Lua folder.
* **QLearning** contains the python scripts that use tensorflow to run and train the neural network. Place that folder wherever you’d like (I recommend not to put it in your Dropbox folder).

## Emulator Setup

You need to get the game running, get the Lua scripts running, and potentially create save states for training.

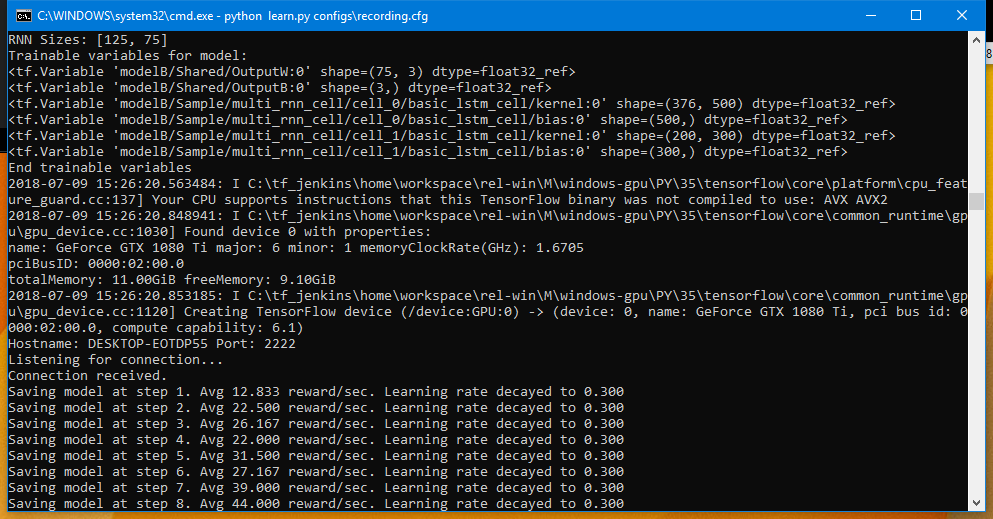
* Open Bizhawk.
* Use File🡆Open ROM to open your Super Mario Kart ROM.
* **TIP**: You can make this ROM autoload every time by using File🡆Recent ROM🡆Autoload
* Use Tools🡆Lua Console to open the Lua console.
* In the Lua Console window, use File🡆Open Session to open **MarIQ/MarIQ.luases**.
* **TIP**: You can make the Lua Console automatically open with Bizhawk by using Settings🡆Autoload in the Lua Console window.
* **TIP**: You can make the Lua console automatically load this session every time by using File🡆Recent Sessions🡆Autoload in the Lua Console window.
* In the main Bizhawk window, use Config🡆Profiles…, and select the Casual Gaming profile. This runs the game the fastest.
* The MarIQ download comes with save states for each course in mushroom cup, but if they don’t work you’ll need to use the provided **Setup.lua** script to create them. To do so, simply double click the Setup script in the Lua console window and let it run until it reboots the game and says it’s done.

# Running the Neural Network

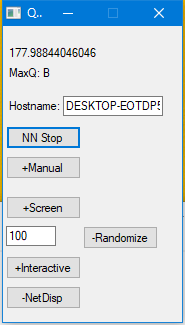
The neural network runs via TensorFlow in a python script, and it connects to a Lua script in the emulator. First you’ll need to run the python script, which will create a TCP server, then you’ll run the Lua script and connect to that server.

## Running the Python Script

* Open a command prompt and navigate to the **QLearning** folder that you extracted in the installation process.
* Use **python learn.py <config filename>** to run the python script.
  + Config files are stored in the **config** folder within the **QLearning** folder.
  + The program defaults to **configs/sample.cfg**.
  + The config file contains all the specifications necessary to define the neural network model and the training procedure.
* After some time (potentially a few minutes), the neural network will be ready to go and the python script will create a TCP server. It will tell you what your computer’s hostname is, which looks something like this.



* **TIP:** To stop the python script, use Ctrl+Break in the command window.
* In the Bizhawk emulator, open Super Mario Kart and the Lua console with MarIQ.luases.
* Run the QPlay Lua script. This should bring up a window that looks like this.



* Change the Hostname field to whatever the python script said your computer’s hostname is.
* Click NN Start. You should see “Connection received.” in the command window.
* Unpause your emulator. The training should being.
* A neural network checkpoint will be saved every 10 in-game minutes by default, and automatically reloaded if you restart the python script.
* You can speed the emulator up for faster training via Config🡆Speed/Skip🡆Speed 400%

## Displaying Extra Information

MarIQ.luases includes several extra scripts to display extra information about what’s going on with the neural network and Q learning process. Simply double click the script name to turn on the display.

| DisplayContoller | Displays the actual buttons being pressed and sent to the game. You can drag the controller image around with the mouse, and use the mouse wheel while the pointer is over the controller image to resize it. |
| --- | --- |
| DrawSmiley | Draws a smiley face that indicates the expected future reward. |
| DrawArrows | Draws an arrow to indicate the expected future reward for each of the first three neural network outputs. This script assumes that they mean straight, right and left in that order. |
| ShowReward | Displays green and red numbers whenever reward or penalty are incurred. |

# Further Options for Exploring MarIQ

The information above is enough to get a neural network to teach itself how to play Super Mario Kart. However, MarIQ is quite configurable, and this section will explain all the ways in which you can explore the different aspects of Q Learning and neural networks with MarIQ.

## Learn.py Command Line Options

The full learn.py usage is:

python learn.py <config file> [--ignore\_checkpoint] [--evaluate]

* The config file should be stored in the **configs** folder.
* Normally if you halt **learn.py** and run it again with the same config, it will reload from the last saved checkpoint. The --ignore\_checkpoint switch will make it ignore that checkpoint and start again from scratch.
* The --evaluate switch puts the neural network into a non-training mode. It will always run at 100% Tryhard, and will never learn from its gameplay. This is useful for evaluating how good the neural network is when it isn’t trying to explore new strategies.

## Learn.py Config Files

A couple of sample config files have been provided. These config files contain a neural network model definition, as well as all the parameters necessary to define the Q Learning training procedure. This section contains an exhaustive list of the config options available.

| Section: **[Train]** | |
| --- | --- |
| SeqLength | The number of time steps to group together for training the recurrent neural network through time. |
| BatchSize | The number of sequences to learn from each training batch. |
| LearningRate | The gradient descent learning rate. |
| LRDecay | The exponent by which to multiply the learning rate at each checkpoint. |
| RewardDiscount | The reinforcement learning reward discount factor to apply at each time step. |
| ExperienceReplays | The number of experience replay sequences to hold onto in memory. |
| CheckpointSeconds | The number of seconds between each checkpoint (when the neural network gets saved to file). |
| PrioritizedExperience-  Replay | If set to “true”, enables Prioritized Experience Replay (which doesn’t seem to work very well in my experience). |
| GradClip | Clips the gradient by a global normalization threshold. |
| Regularization | The coefficient for a regularization term in the training cost. |
| Section: **[Model]** | |
| StepsPerSecond | The number of time steps the neural network processes per second. |
| DoubleDQN | If set to ‘true’, enables Double DQN (two neural networks to increase learning stability). |
| Section: **[RNN]** | |
| Layer*N* | Starting with Layer1, the number of LSTM cells per layer of the Recurrent Neural Network. |
| Section: **[Checkpoint]** | |
| Dir | The directory where the checkpoint data will be stored. **This should be different for every config file**. Also make sure it isn’t in a dropbox directory. |
| Section: **[Emulator]** | |
| seconds\_per\_course | The maximum number of seconds to stay on each course. |
| exploration\_softmax | A coefficient used in for a softmax selection of the buttons used for exploration. |
| tryhard\_min | The minimum Tryhard percentage (percentage of time steps in which the optimal action is taken, rather than exploration). |
| track\_position | If set to 1, kart position and rotation is stored with the experience replays. This can be used to generate multikart videos. |
| Section: **[Controller]** | |
| Order | The order in which controller buttons will be received from the emulator. |
| Buttons*N* | Starting with Buttons1, each line defines a single neural network output as a set of buttons to press. |

## QPlay Lua Script

The QPlay Lua script creates a window that can be used to play around with MarIQ’s training.

* NN Start/Stop can be used to connect and disconnect from the python script.
* +Manual/-Manual can be used to enable manual driving. The Q learning algorithm can learn from your driving, though it may have unintended side effects…
* +Screen/-Screen displays the neural network’s view of the game.
* +Randomize/-Randomize enables and disables Tryhard randomization. You can manually set a Tryhard value in the text field next to it.
* +Interactive/-Interactive allows you to play alongside the neural network. The game will periodically swap who has control of the kart.
* +NetDisp/-NetDisp enables and disables the pygame window that displays neural network information.