Required Programs and Links to Get Them

- DesMuMe (http://desmume.org/download/): The Nintendo DS Emulator I used for this project.
- ROM file for Mario Kart DS (https://www.romsgames.net/nintendo-ds-rom-mario-kart-ds/): The file that I used to get the game on the emulator
- Lua 51.dll compiler (https://sourceforge.net/projects/luabinaries/files/5.1.5/Tools%20Executables/lua-5.1.5_Win64_bin.zip/download): How you get lua script working on DesMuMe, as it needs a compiler.

How to get it Working*

*This guide is for getting it to work on a Windows computer. Getting it to work on Mac or Linux is most likely the same but it could possibly be slightly different

1. Put DeSmuMe and the LuaScript compiler all in the same directory (I put both of them in a folder on my Desktop).
2. Launch DeSmuMe, then go to File (in the top left, like most programs), then go to "Open ROM", and click on the zipped folder of your Mario Kart DS ROM. Select Single-Player Mode on the title screen of the game.
3. Back To DeSmuMe tinkering, go to Tools, then to Lua Scripting, and a window should pop up. From there click "Browse", and go to the Lua Script file that you downloaded from this repository (doesn't matter what directory the script file is in, just the compiler). If it is not working you might consider building Lua 5.1 on your PC (lua-users.org/wiki/BuildingLuaInWindowsForNewbies)
4. Now, start the Python Code on PyCharm (what I used) or any other Python IDE that suits you best. Wait for it to display the summary of both the Q-networks. Make sure you have all libraries that the code used installed.
5. Go back to Mario Kart and select the Vs. mode. You can select any character you want (I went with R.O.B. because he is a robot, heavy, And with a high speed stat, mostly because he's a robot, but you have to unlock him through playing the game yourself and beating

Special Cup on Mirror Mode, so if that's not your style, you can pick from 8 other characters, including Toad, Yoshi, Luigi, and Donkey Kong), but make sure to set it to 150cc (to make training quicker) and on easy difficulty (to make training more forgiving) in the nest screen that follows after you select your character. Press "Okay".

6. Once you get the list of courses pick the Mushroom icon then select Figure 8 Circuit, then press "Okay". From this point on you don't need to do anything! You can just watch it train! You might want to use the PreventTimeOut program in the repository as well depending on your computer's resources. I picked Figure-8 Circuit as a Track because

How it Works

*Also detailed in the code files, but here is a compilation of all the explanations I give. More in-depth ones for each program can be found on their respective code files.

The network is a dynamic convolutional neural network that takes in 6 frames of screenshots covering roughly the area of the DS frame (both top and bottom, used to be just bottom then it was reworked to be both in fear of it not knowing what to do when parts of the track intersect on the minimap) in DeSmuMe and then chooses one out of 5 actions to take (turning left, turning right, going straight, and the former three with the addition of using an item, all of which controlled by keyboard) using a Conv3D architecture (Conv2D was too computer-intensive and did not learn too much), getting a reward based on direction and speed. When I first made the program, I also gave it the capability to drift, but I scrapped that very early, as it was making the robot too chaotic. However, I added it back in the final code as a possible option to enable if you want to see what happens. The Lua Script is the main rewarder of the bot, as it makes a box with certain g and b values (RGB color) depending on speed and direction (defined relative to checkpoint it most recently passed), both values it gets by reading the memory of the game, something the Python code cannot do, on the top left side of the bottom screen, which the Python program reads the RGB values from and gets the reward. The higher speed, the higher the absolute value of the reward, but whether it's going in the right direction or not determines the sign of that reward (wrong direction is negative reward,

right direction is positive reward). The network follows a DDQN training algorithm, with some extra things I had to put in there. Namely, I had to make it where it only remembers the last 10 races because the rate of training would slow massively if I didn't do so (my GPU is bad), and I had it forget the last two sets of frames, which were between the time it finished and the time that the program says its finished. Additionally, it forgets fairly quickly any races where it thinks it's racing although it isn't even on the race screen. Although, enough time when it thinks its racing although it isn't can still seriously detriment or completely impair your program from learning the right thing. Mario Kart DS's DNF system was very helpful, as that would tell the bot the race was over without it actually being able TO finish the race, so it doesn't spend an eternity in one race. The network only starts training once the race finishes, and then when it finishes training on that batch for an episode, it goes to the next race on its own through a series of keyboard presses.

Expected Results

The training video shown was at episode 804, where it went to the final lap. This does show its progress, as trackwise, by the 600th/700th episode (out of 2000) your bot should be consistently clearing the first lap before it gets DNFed. It takes it a while to learn how to do it, but don't fret!. It does eventually learn.

<u>Pictures to look at Score-wise Expected Results</u>
Average score is the statistic that should be replicable, score not so much: