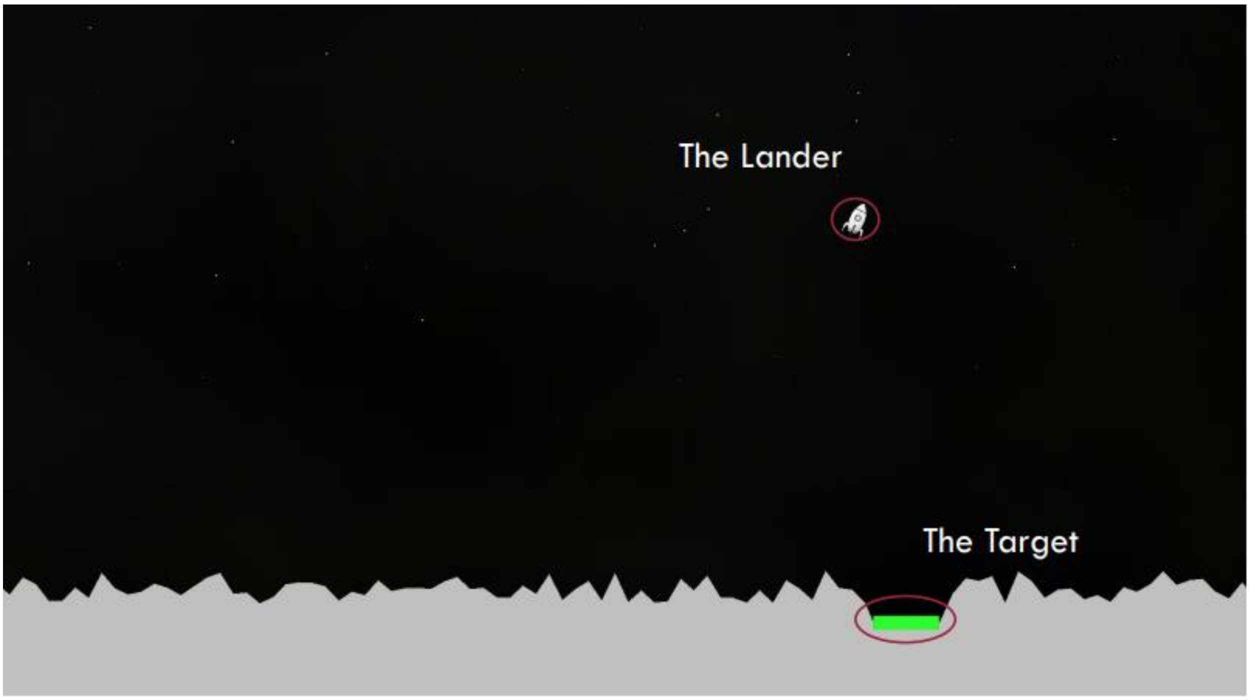
Neural Network Built from Scratch to play the Lunar Lander Game

The objective of the game:

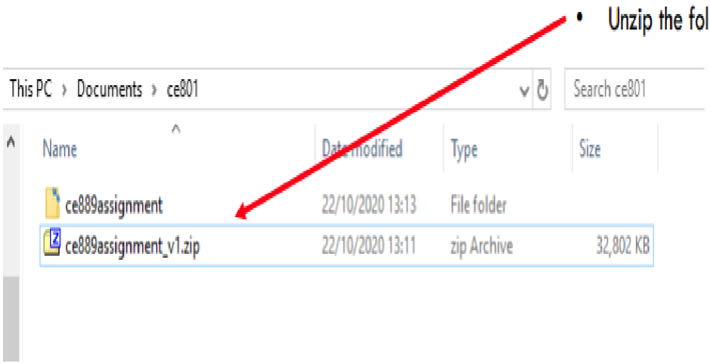
- Steer and apply thrust to the lander.
- Avoid hitting the outside edge or the ground.
- Safely put the lander on the target to proceed.



Objectives of the project:

- In the lunar lander game, the user controls a spaceship that is trying to land on a specific target area of the map. The user needs to move the spaceship towards the target area and then slowly move it down so that it lands correctly.
- The objective is to l design and implement a neural network with a single hidden layer that will be able to play the lunar lander game simulator.
- The neural network should be implemented in python no external libraries will be used.
- The neural network will receive two inputs (distance to target in X and distance to target in Y) and predict what should be the expected velocity in X and in Y (two outputs).
- The game simulator of the lunar lander game was provided. The focus of the project will be on designing and implementing the neural network.

Running the game:

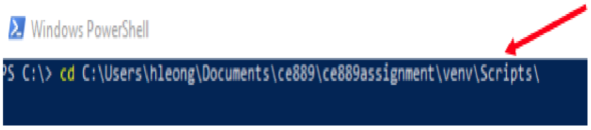


This PC › Documents › ce801

Search ce801

Name	Date modified	Type	Size
ce889assignment	22/10/2020 13:13	File folder	
ce889assignment_v1.zip	22/10/2020 13:11	zip Archive	32,802 KB

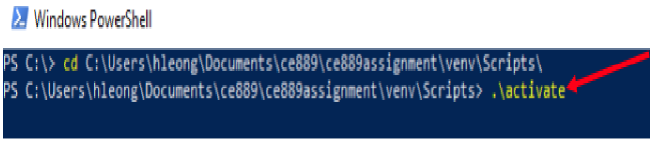
- Unzip the folder



Windows PowerShell

```
PS C:\> cd C:\Users\hleong\Documents\ce889\ce889assignment\venv\Scripts\
```

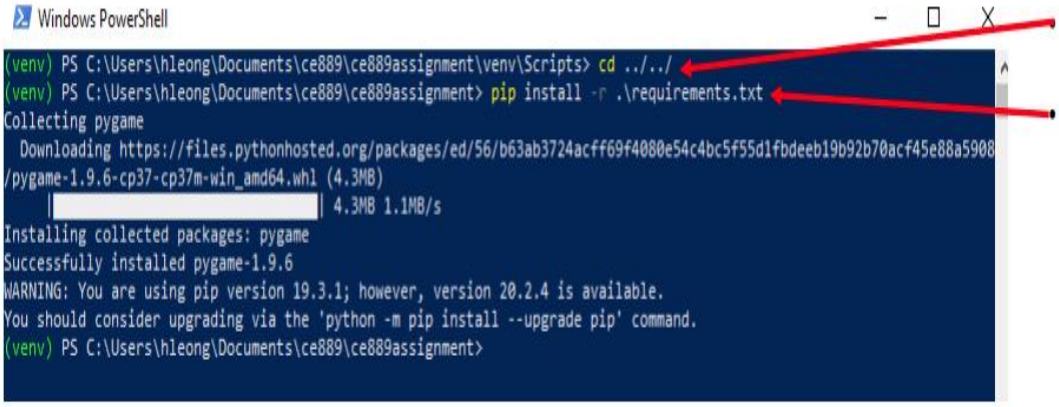
- Open powershell
- Change directory to the folder venv/Scripts/ inside the folder you just unzip



Windows PowerShell

```
PS C:\> cd C:\Users\hleong\Documents\ce889\ce889assignment\venv\Scripts\
PS C:\Users\hleong\Documents\ce889\ce889assignment\venv\Scripts> .\activate
```

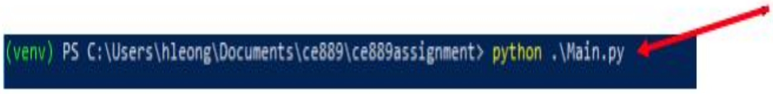
Activate the python virtual environment



Windows PowerShell

```
(venv) PS C:\Users\hleong\Documents\ce889\ce889assignment\venv\Scripts> cd ../../
(venv) PS C:\Users\hleong\Documents\ce889\ce889assignment> pip install -r .\requirements.txt
Collecting pygame
  Downloading https://files.pythonhosted.org/packages/ed/56/b63ab3724acff69f4080e54c4bc5f55d1fbdeeb19b92b70acf45e88a5908
/pygame-1.9.6-cp37-cp37m-win_amd64.whl (4.3MB)
     |████████████████████████████████| 4.3MB 1.1MB/s
Installing collected packages: pygame
Successfully installed pygame-1.9.6
WARNING: You are using pip version 19.3.1; however, version 20.2.4 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
(venv) PS C:\Users\hleong\Documents\ce889\ce889assignment>
```

- Go back to the main folder of the project
- Install requirements

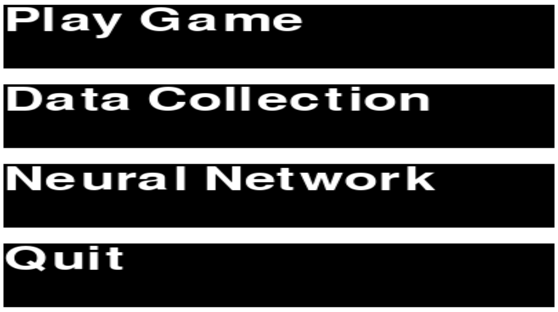


```
(venv) PS C:\Users\hleong\Documents\ce889\ce889assignment> python .\Main.py
```

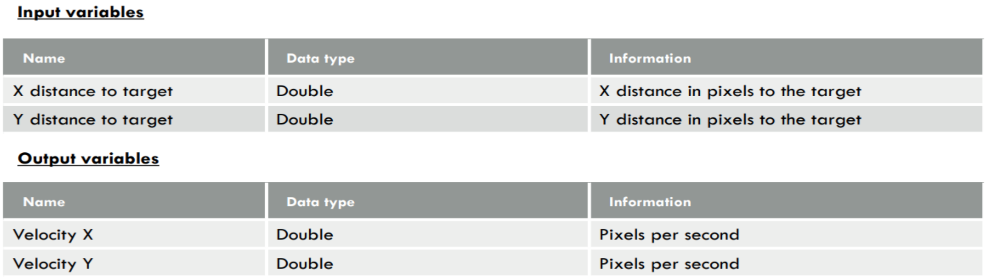
Run the project by executing Main.py

Data Collection:

- After opening the game folder, we will be able to see the following options



- Here the Data Collection will allow us to play the game many times and record the following data.



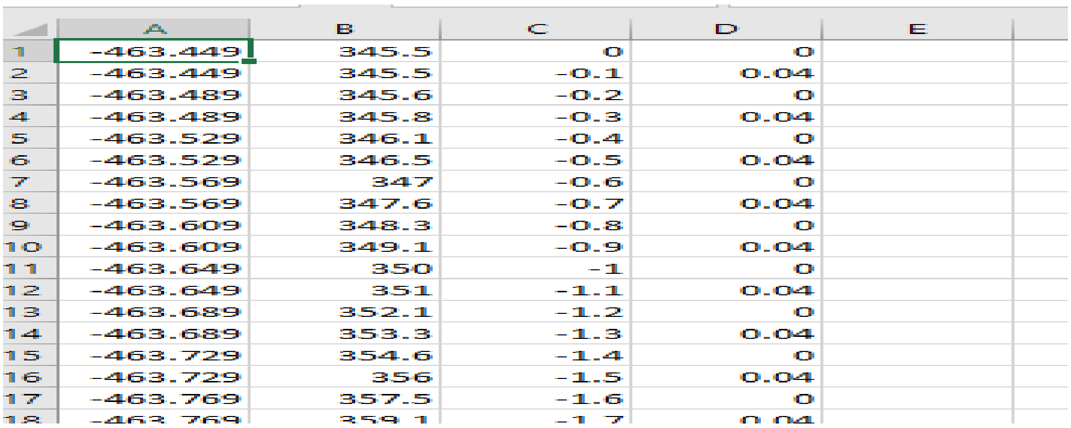
Input variables

Name	Data type	Information
X distance to target	Double	X distance in pixels to the target
Y distance to target	Double	Y distance in pixels to the target

Output variables

Name	Data type	Information
Velocity X	Double	Pixels per second
Velocity Y	Double	Pixels per second

- Each run of the game is added to the same file once the game has been closed. The data in output to the ce889_dataCollection.csv file



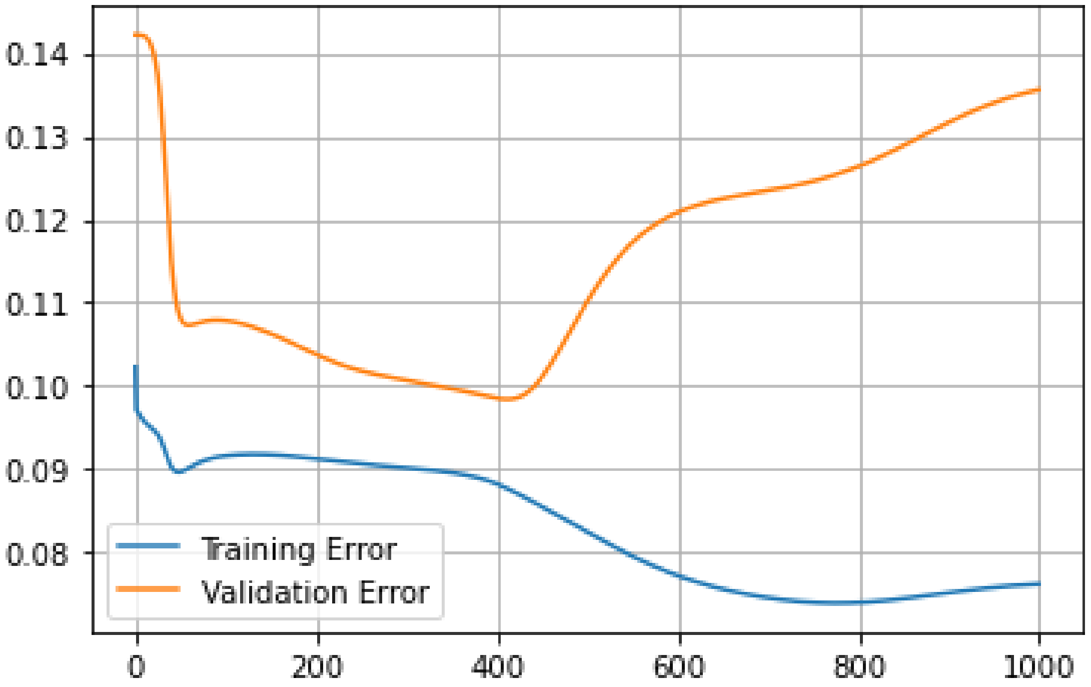
	A	B	C	D	E
1	-463.449	345.5	0	0	
2	-463.449	345.5	-0.1	0.04	
3	-463.489	345.6	-0.2	0	
4	-463.489	345.8	-0.3	0.04	
5	-463.529	346.1	-0.4	0	
6	-463.529	346.5	-0.5	0.04	
7	-463.569	347	-0.6	0	
8	-463.569	347.6	-0.7	0.04	
9	-463.609	348.3	-0.8	0	
10	-463.609	349.1	-0.9	0.04	
11	-463.649	350	-1	0	
12	-463.649	351	-1.1	0.04	
13	-463.689	352.1	-1.2	0	
14	-463.689	353.3	-1.3	0.04	
15	-463.729	354.6	-1.4	0	
16	-463.729	356	-1.5	0.04	
17	-463.769	357.5	-1.6	0	
18	-463.769	359.1	-1.7	0.04	

Data Pre-processing:

- Normalisation All data need to be scaled between 0-1. This was done using min-max scaling
- Any inconsistent data was removed or replaced
- The data was then split into train and validation, where 70% of the total data was used for training and the rest for validation

Training the model:

- A Neural network with single hidden layer was designed for training the model.
- All the mathematical formulas for feed-forward and back propagation was implemented from scratch.
- The model was trained for a total of 2000 epochs and the following was the result.



- It was observed that the model gave least validation error at 405th epoch, so the waits at that epoch was saved and loaded to the game.

Implementing the outputs in the game:

- The final output which was obtained by NeuralNetHolder.py was scaled back to the original values
- After various trial and error, it was found that there was an error equivalent to 0.65 pixels in the output, so this error was added to the input rows.