**Test 2 Part 1 – The Pong Analysis Report**

Albar Murtuza Vasi

C0854302

CBD2204\_3 - Big Data Strategies

Cloud Computing for Big Data, Lambton College of Toronto

Professor Graham Wall

15th August,2022

Table of Contents

**GitHub**3

**Introduction**3

**Work Done**3

Azure Virtual Machine3

Python Code and Libraries7

Threshold, Neurons and Learning Rate10

Table 1 – Learning based on Neurons11

Graph 1 – Neurons vs Execution Time, RAM and CPU18

Table 2 – Learning based on Learning Rate19

Graph 2 – Learning Rate vs Execution Time, RAM and CPU28

**Conclusion**28

**References**29

**GitHub**

Link - <https://github.com/albarvasi/Pong_Analysis_Test2_Part1>

**Introduction**

In this Analysis experiment, the students were tasked to run a Ping Pong code, and do analysis by changing the number of neurons and learning rate.

Neurons – The number of brain cells given to the AI

Learning Rate – The rate at which the AI learns to win the games.

Just to provide an overall summary of the game, this game has been implemented with reinforced learning, to train and teach the AI to win games against a user. Each episode has a set of 21 games and based on the number of games the AI wins the total reward is calculated.

If the reward is -21, it emphasizes that the AI lost all the games. If the reward is -17, it underlines that the AI won 4 games in that particular episode.

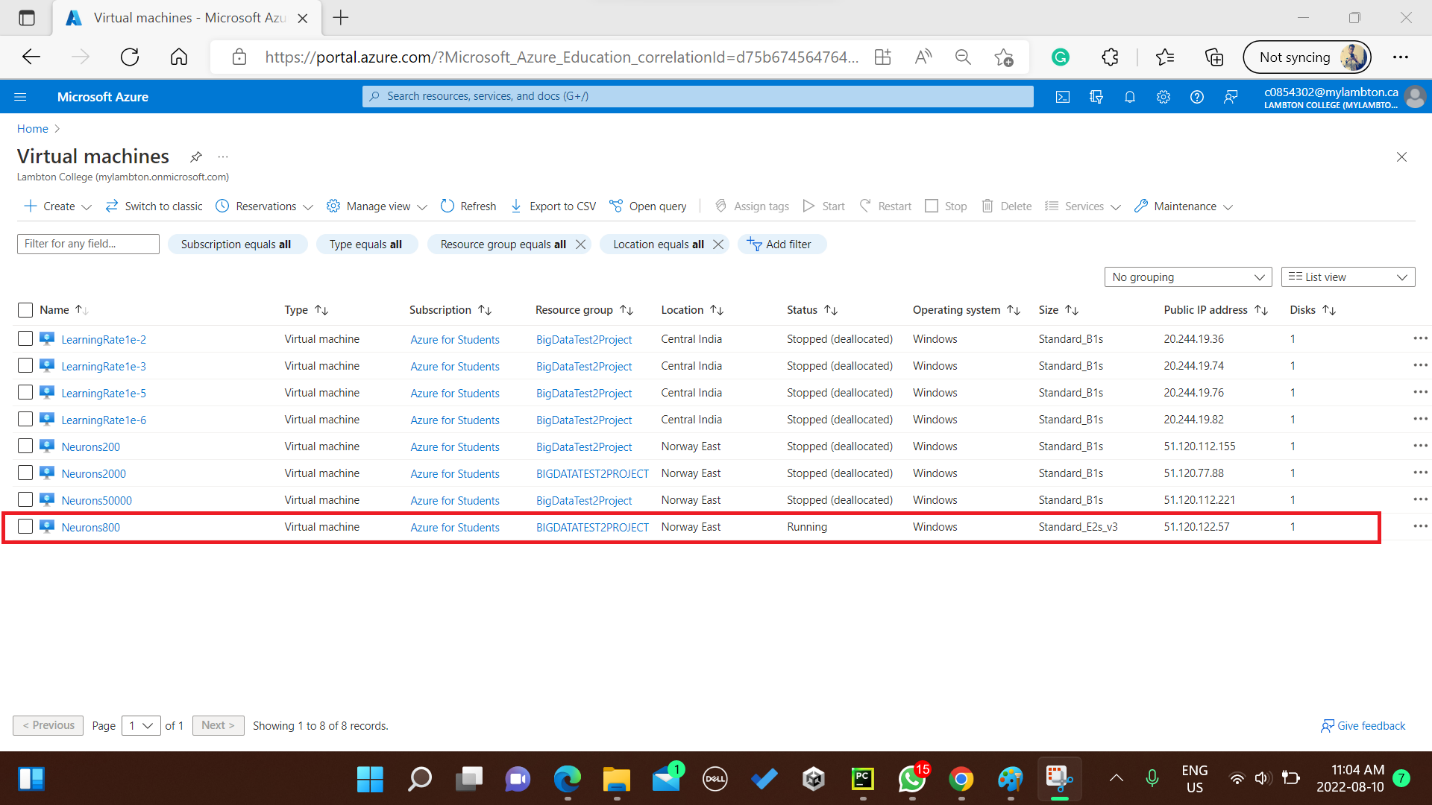
Students were also instructed that they have the option to alter the code to meet their threshold and data logging requirements.

**Work Done – Azure Virtual Machine**

For this analysis experiment, I created Azure VMs to complete multiple iterations together, using the student credits provided by Lambton College in Toronto in the process.

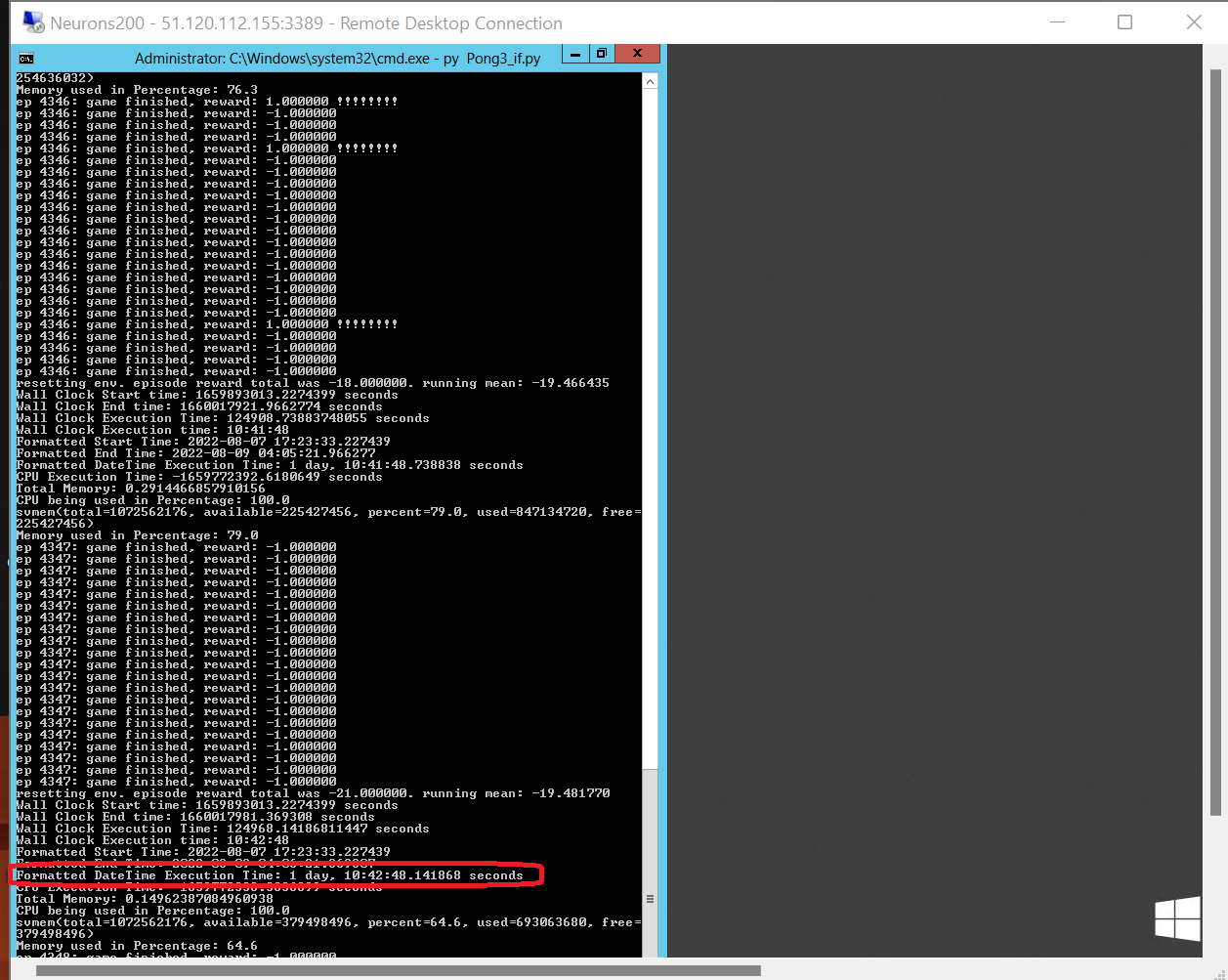
I started by created eight Windows Server 2012 machines with selecting size as Standard\_B1s which is general purpose and has 1 vCPU and 1 GiB RAM. The chosen storage disk was Standard SSD.

The reason for choosing Windows Sever 2012 was that it is a light weight operating system and does not uses a lot of RAM.



*Fig 1 – All the Virtual Machines created initially (own photo)*

The speed of the code running was not so efficient on 1 vCPU and 1 RAM, and the results were not converging, i.e., the AI was not winning many of the games even after 2 days of running continuously.

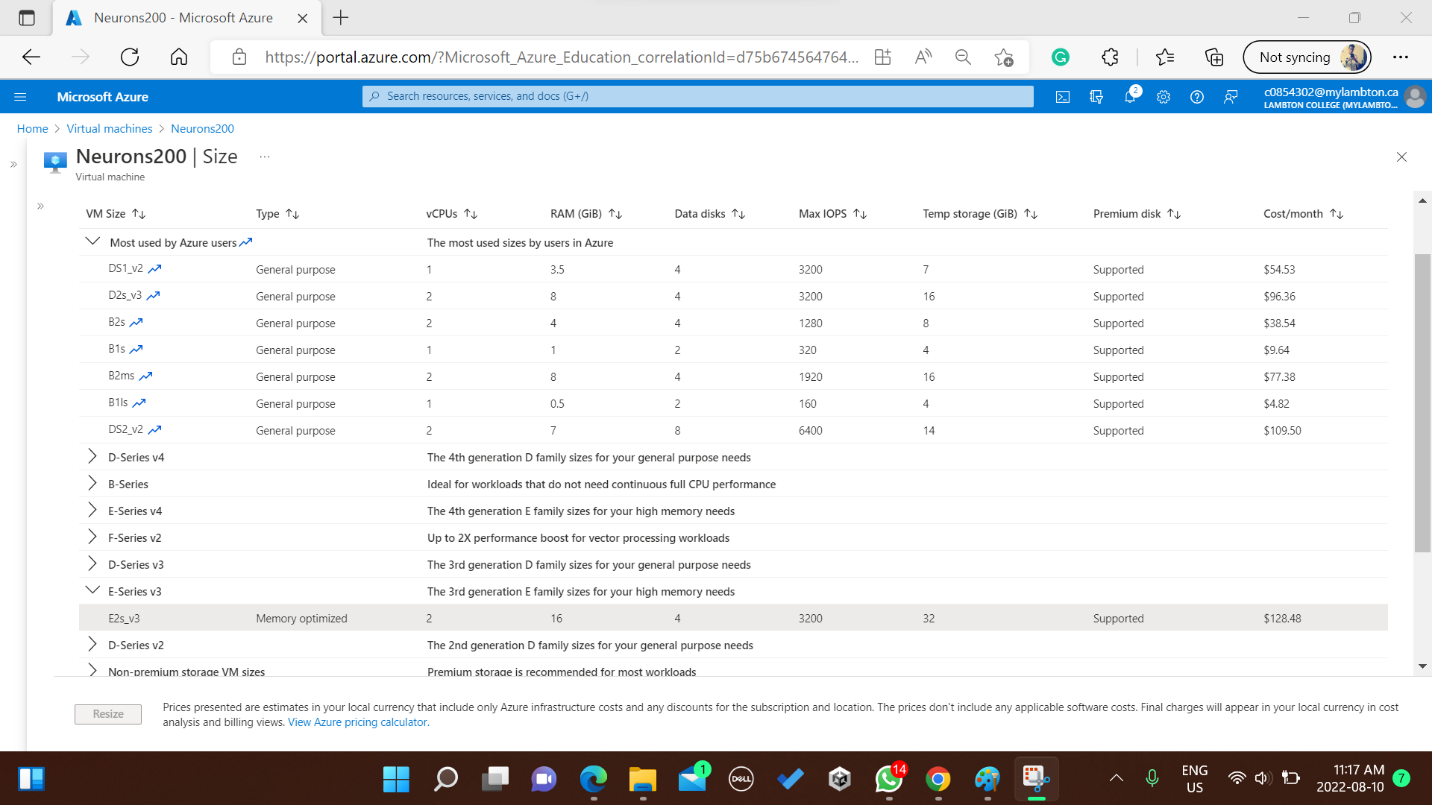


*Fig 2 – Neurons 200 with Base Learning rate code on 1vCPU and 1 RAM(own photo)*

As you can see in Fig 2, even after 1 day and 10 hours, the running mean was just -19 and this was for just the base code with 200 neurons and learning rate as 1e-4. This displayed that, if I ran all my iterations on 1 vCPU and 1 RAM, I would be able to complete maybe some iterations if and only if the learning rate is very high. It seemed too difficult to conduct the whole experiment on time on basic learning rate.

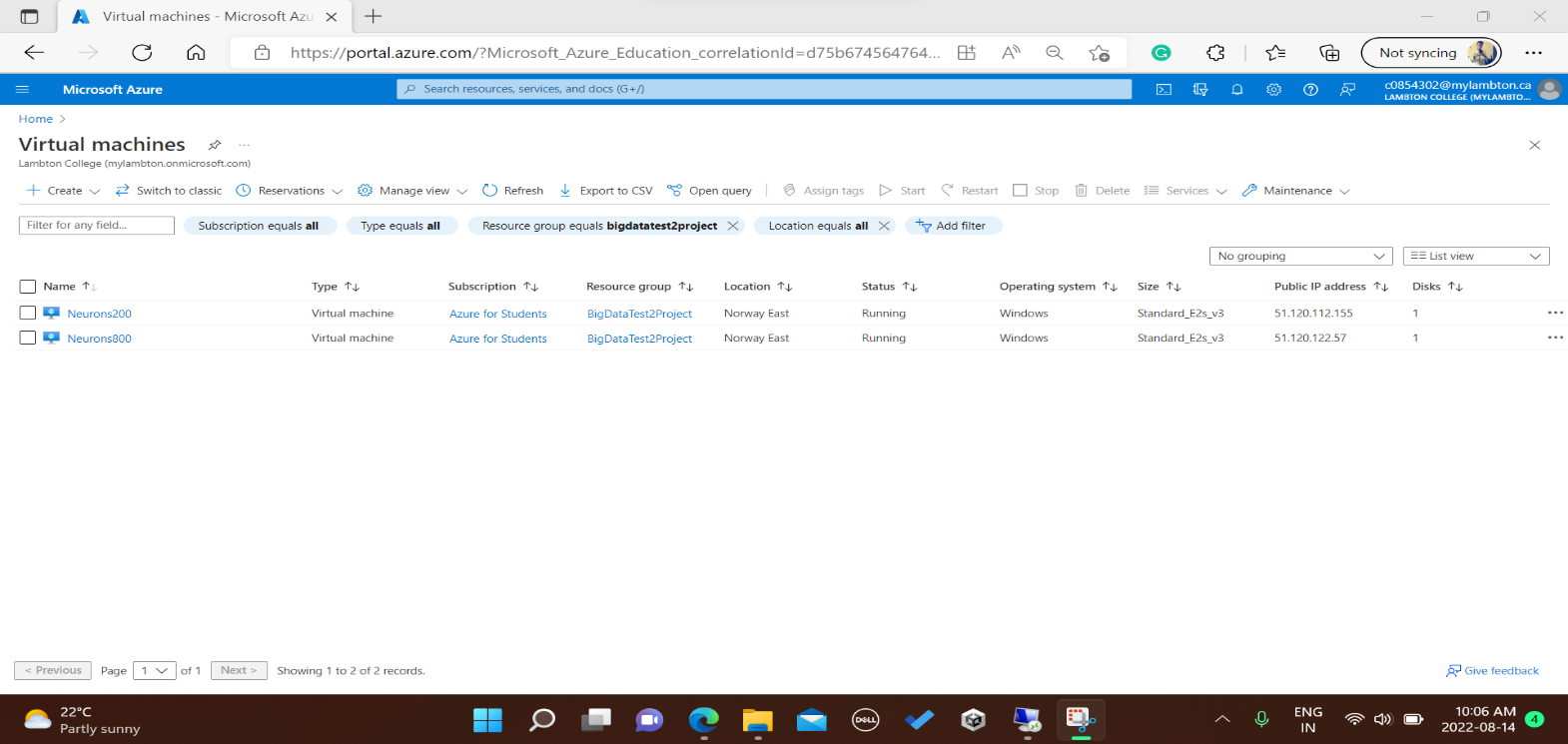
After confirming with professor about using more than 1 vCPU, I decided to resize the VMs, with 2 vCPUS and 16 GIB RAM.

I resized the VMs using Standard\_E2s\_V3 with standard SSD. I have marked the same in **Fig 1** as well.



*Fig 3 – Resizing VM for more vCPU and RAM(own photo)*

The execution and performance time increased considerably, it was conclusive that running 8 VMs simultaneously will not be required, and therefore resized another one of the VMs and deleted the rest 6 VMs. The following screenshot displays how my azure virtual machine looked after resizing the VM.



*Fig 4 – After Resizing and deleting unnecessary VMs[own photo]*

The reason for using Azure Virtual Machine instead of running the code in local computer was to ensure that the local machine was not overwhelmed with the information processing.

**Work Done – Python Code & Libraries**

Apart from the original version of the python code the professor shared with the class, an upgraded version of the python code is also available.

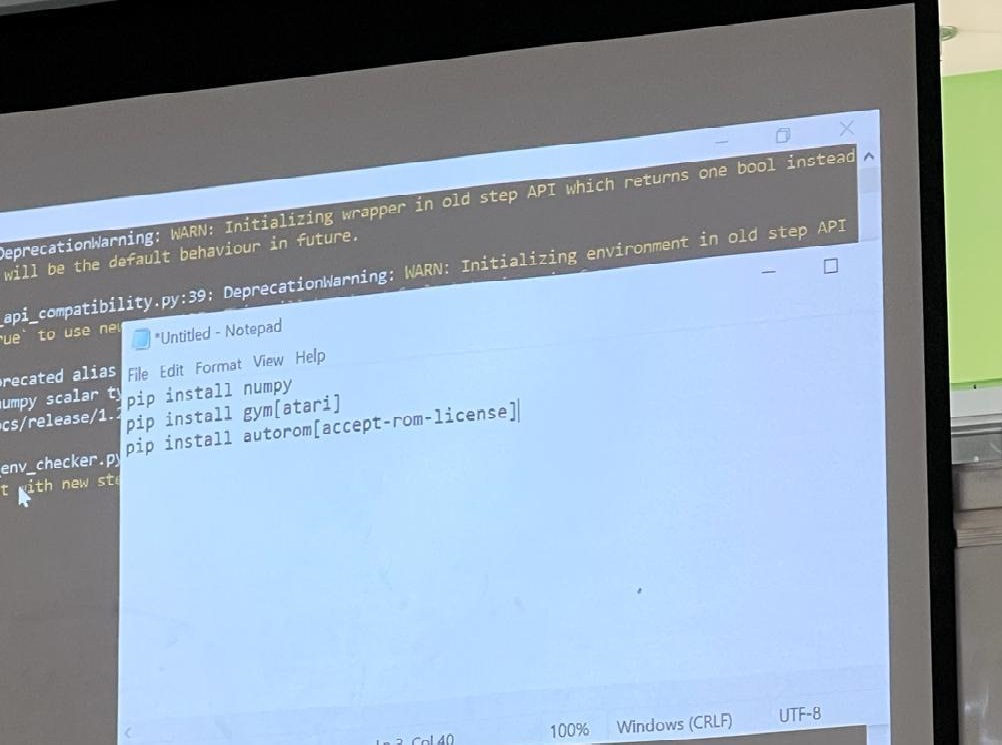
Original code - <https://gist.github.com/karpathy/a4166c7fe253700972fcbc77e4ea32c5>

Upgraded Code - <https://colab.research.google.com/drive/1KZeGjxS7OUHKotsuyoT0DtxVzKaUIq4B?usp=sharing>

The significant differences in both the codes –

* In the original code, the Pong version is 0, i.e., the base Pong version while in the upgraded code the Pong version is 5. The version 5 increased the execution time of the code.
* In the original code, every game played in each episode is printed. In contrast, in the upgraded code, the games are played in the backend and only the episode number with the final result is displayed, reducing the time complexity in the upgraded code.

To make sure that either of the code runs as expected, there was a requirement to install three libraries, i.e., numpy, gym and autorom. As my implementation was done using command prompt or cmd, I had to used the command **pip install library\_name** to install libraries**.**



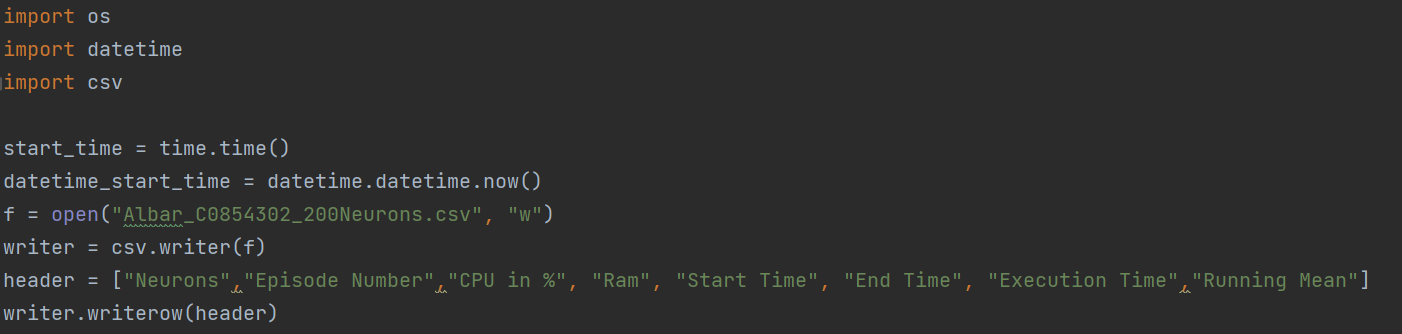
*Fig 5 – Libraries to install before running the python code[own photo]*

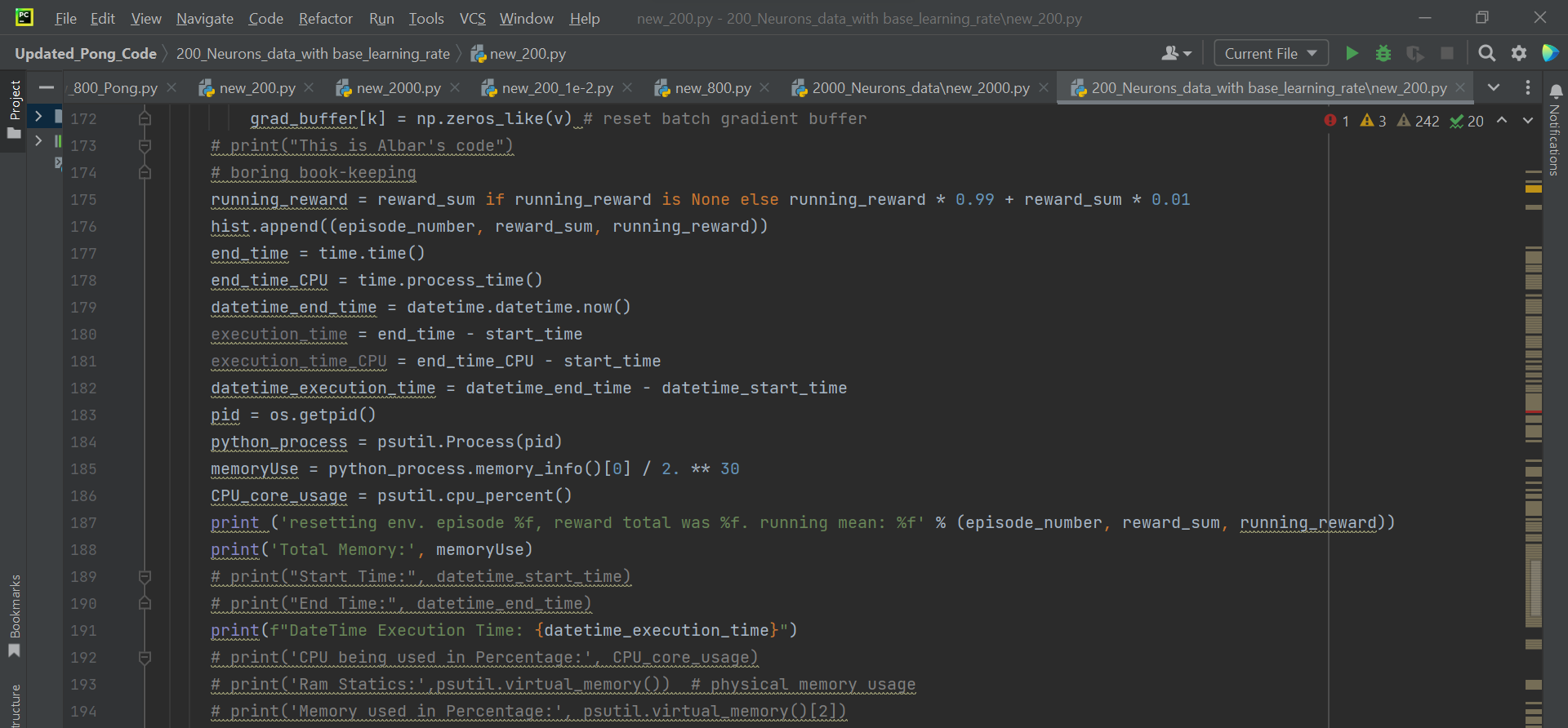
Apart from the libraries shown in **Fig 5,** I also had to install **psutil** – to calculate cpu and ram usage and **date** and **time** libraries – for the start time, end time and execution time.

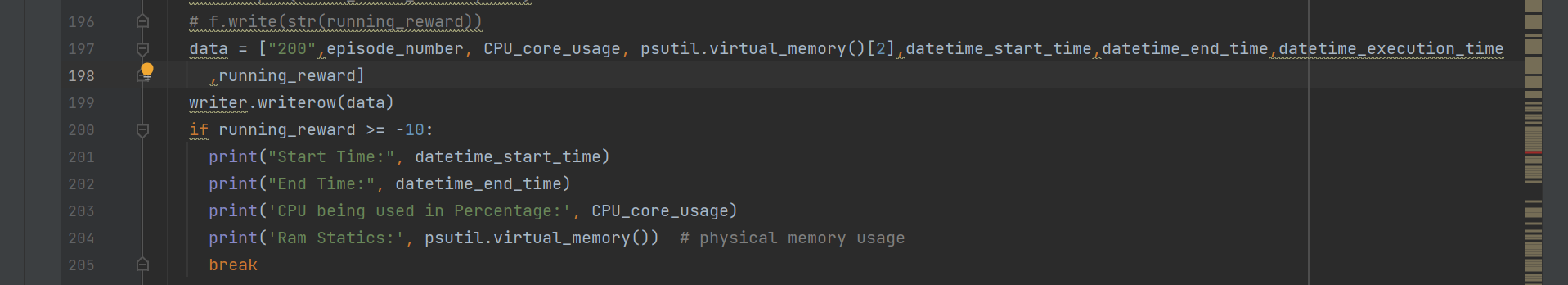
Another thing to note that, the original code was implemented in python 2. To run that code, I had to change the version of code from python2 to python3 using this online tool/website - [Python 2 to 3 convertor online](https://python2to3.com/).

For my analysis, I have implemented the upgraded version of the code and have made significant changes in the code, i.e., setting up the threshold (after how much learning the code should stop), to calculate CPU usage in percentage, to display the memory being used, to display RAM usage in percent, start time, end time, execution time, and saving all this information per episode in an excel file.

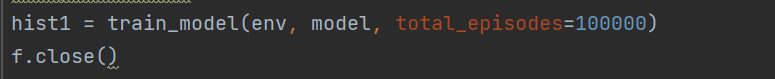
Following are the snippets/screenshots of the changes done in the code –

1 - 

2 -

3 -

4 -



**Work Done – Threshold, Neurons and Learning Rate**

It was emphasized by the professor to ensure that a threshold is set to end the code as the code has a while loop that can keep the code running forever. Threshold could be placed on either the running mean, i.e., the number of games the AI is now winning or the number of episodes.

I have implemented a threshold on both of these parameters.

For running mean, if the running mean hits more than **-10**, the code will automatically stopped, whereas for episodes, if the episode number hits **100,000 or 100K**, the code will stop automatically.

Based on the completed analysis, 6 out of 9 iterations of code stopped after the running mean went over -10, and 1 out of 9 iterations stopped after hitting 100,000 episodes.

The amount of time taken, CPU and RAM usage, has been different for each iteration and these factors depend upon the number of neurons and learning rate.

For my analysis, I used 5 different neuron values i.e., **200, 800, 2000, 10000, and 200000**. 200, 800 and 2000 - completed the running mean threshold, where as 200,000 crashed after 1 episode due to memory issue.

For 10,000 neurons, the windows server 2012, went for updates interrupting that iteration and thus was partially completed. However, the running mean for 10,000 neurons reached -11, so it is safe to say that it would have hit the threshold if windows server 2012 had not interrupted the iteration.

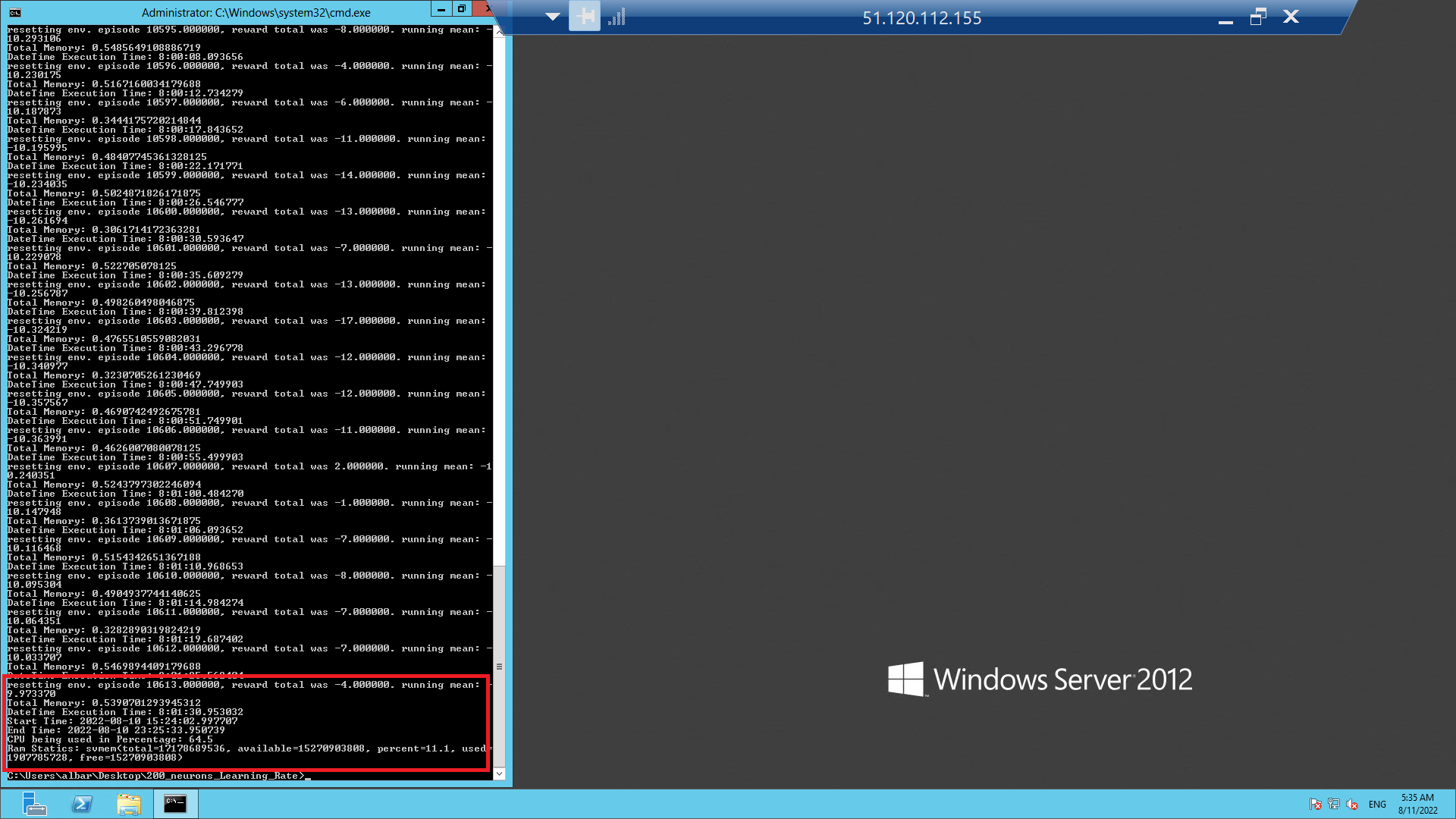
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Iteration | Number of Neurons | Start Time (Time) | Stop Time (Time) | Time Taken (second) | RAM Used | Episodes | Running Mean | CPU |
| 1 | 200 | 10th Aug, 3:24:03 PM | 10th Aug, 11:25:34 PM | 28,891 seconds | 11.1% | 10613 | -9.973370052 | 64% |
| 2 | 800 | 10th Aug,  3:39:32 AM | 10th Aug,  2:29:57 PM | 39,026  seconds | 12.6% | 4798 | -9.911575482 | 87.5% |
| 3 | 2000 | 10th Aug,  3:12:47 PM | 11th Aug,  6:23:08 AM | 54,620.99 seconds | 12.5% | 2902 | -9.976990256 | 94.2% |
| 4 | 10000 | 12th Aug,  2:45:10 PM | 14th Aug,  2:01:59 PM | 170,209.09 seconds | 25.1% | 1493 | -11.61015998 | 97.5% |
| 5 | 200,000 | 15th Aug,  5:24:19 AM | 15th Aug,  5:45:59 AM | 1300 seconds | 98% | 1 | -21 | 77.6% |

***Table 1: Learning based on Neurons***

**Neuron Based Iterations –**

200 Neurons:

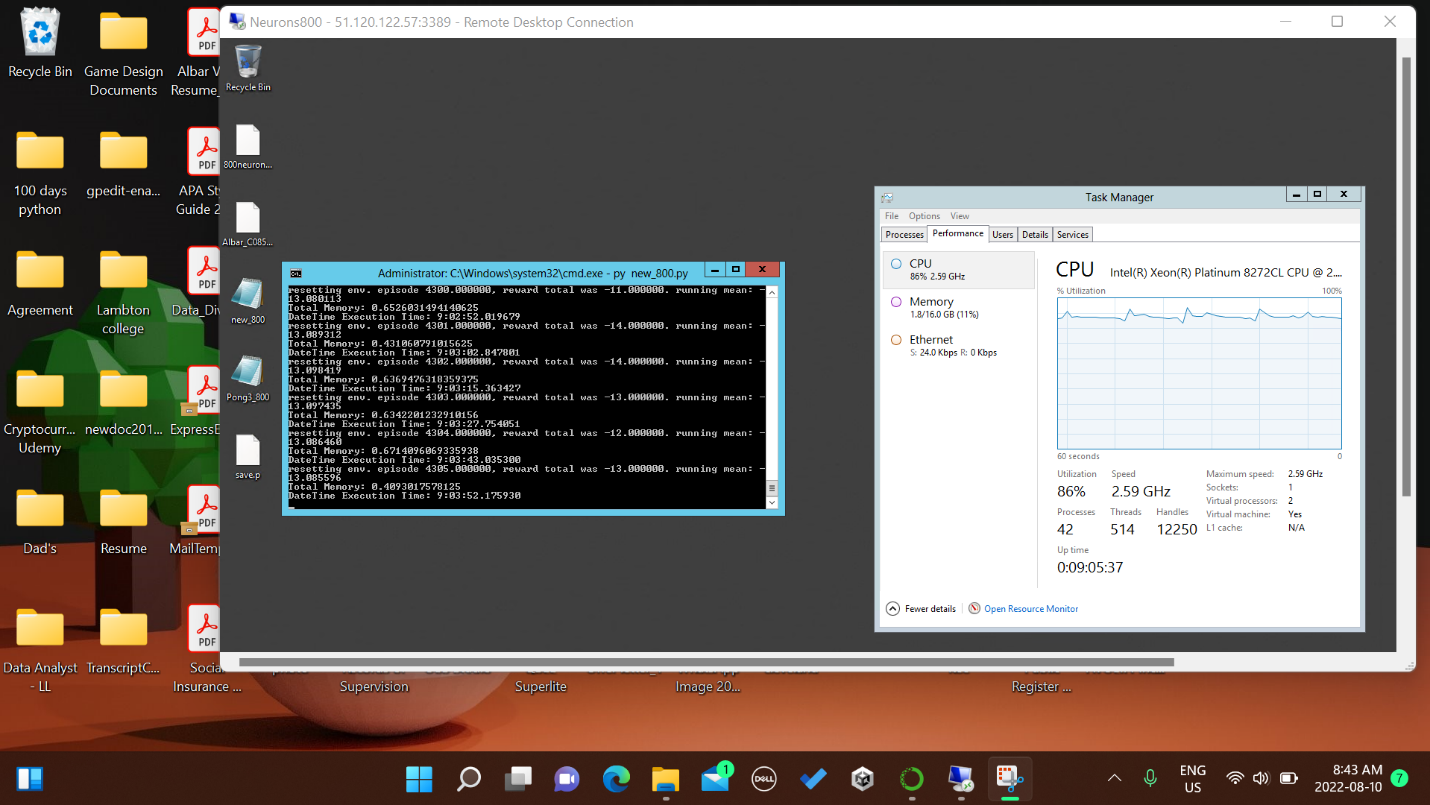
To execute the code for 200 neurons and hit the running mean threshold of greater than -10, it took 8 hours 1 minute and 31 seconds i.e., 28,891.



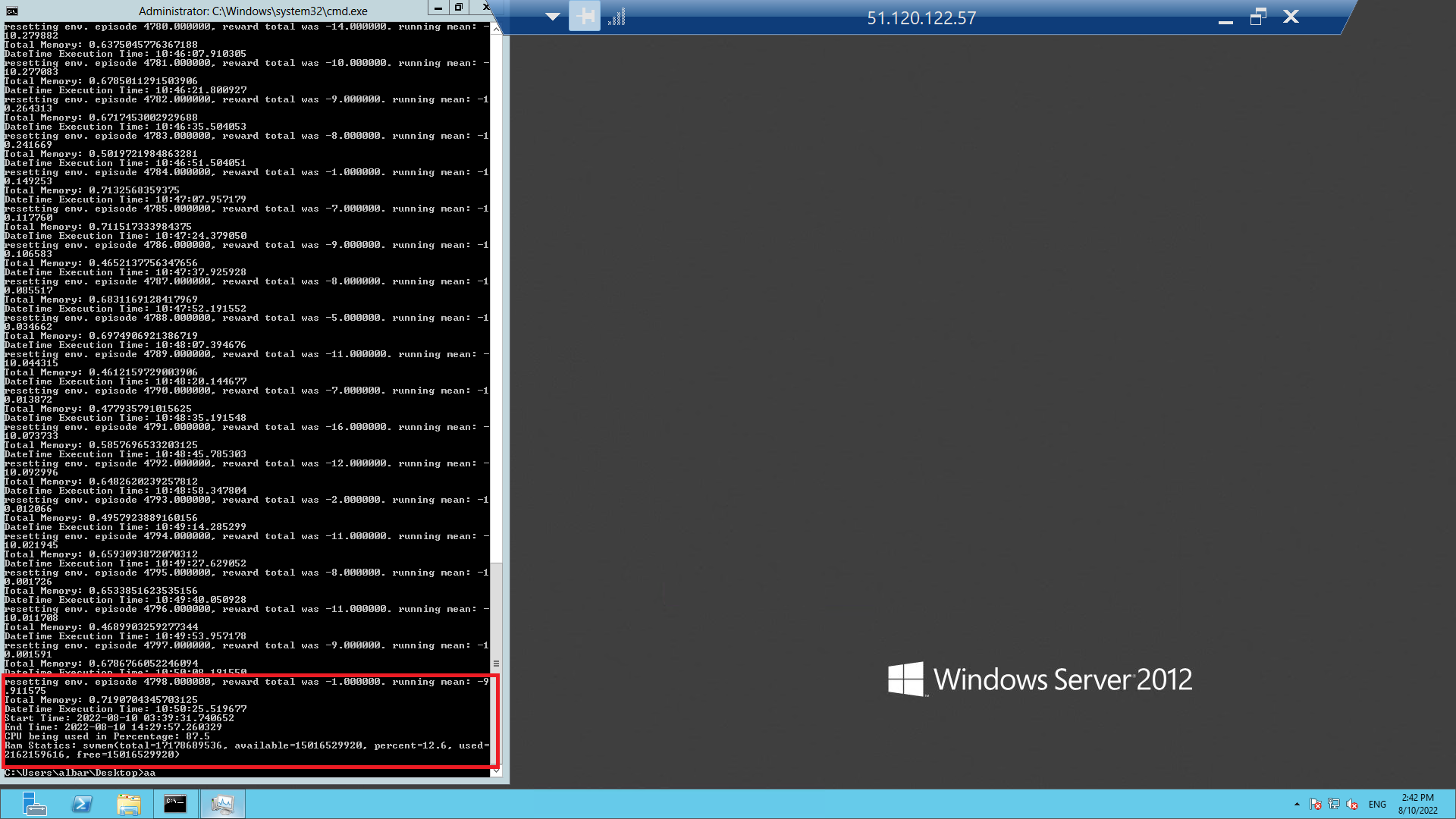
*Fig 6 – 200 neurons termination screenshot (own photo)*

800 Neurons:

To execute the code for 800 neurons and hit the running mean threshold of greater than -10, it took 10 hours 56 minute and 26 seconds i.e., 39,026 seconds.



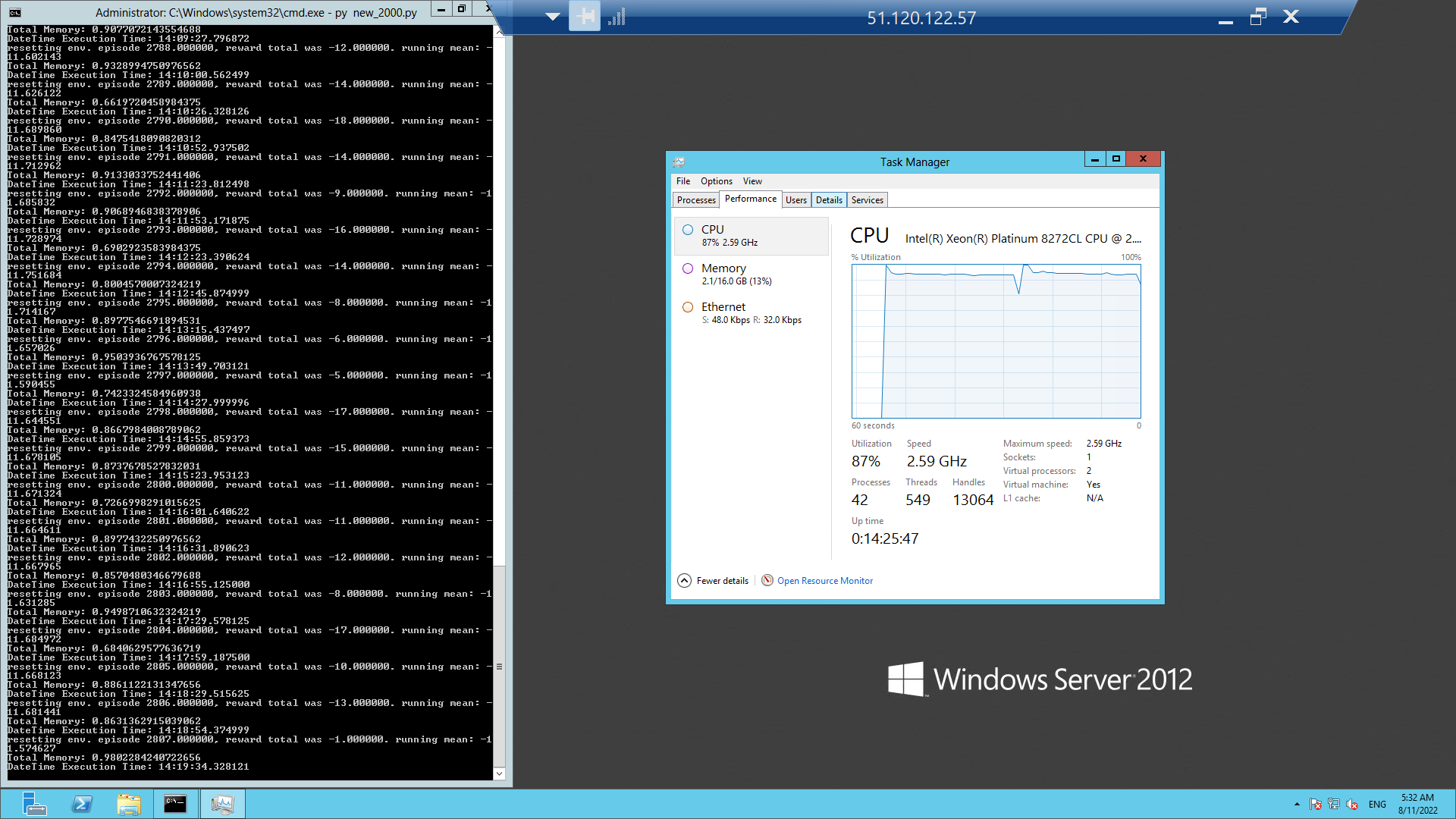
*Fig 7 – 800 neurons after 4000 episodes*



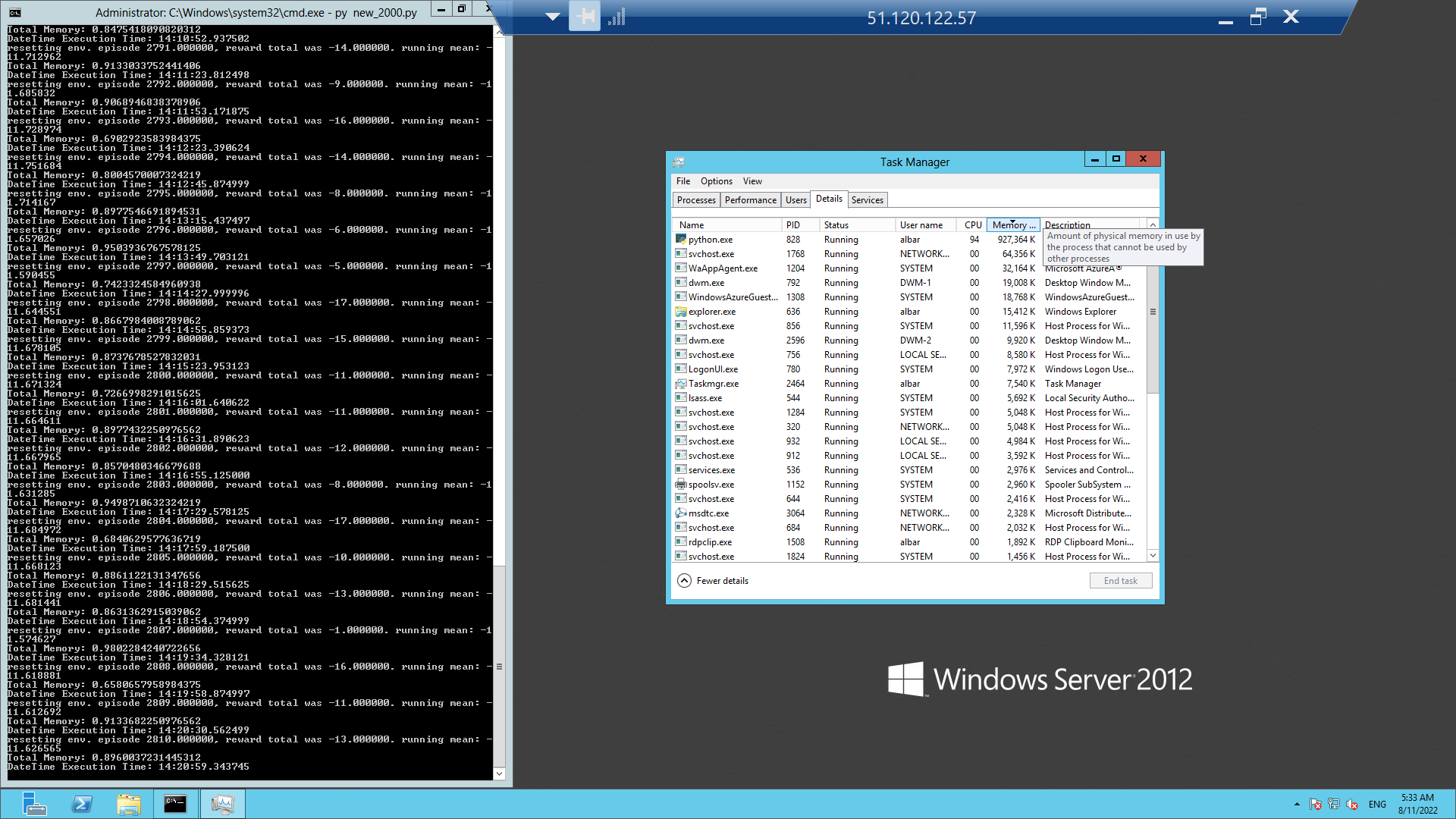
*Fig 8 – 800 neurons termination screenshot*

2000 Neurons:

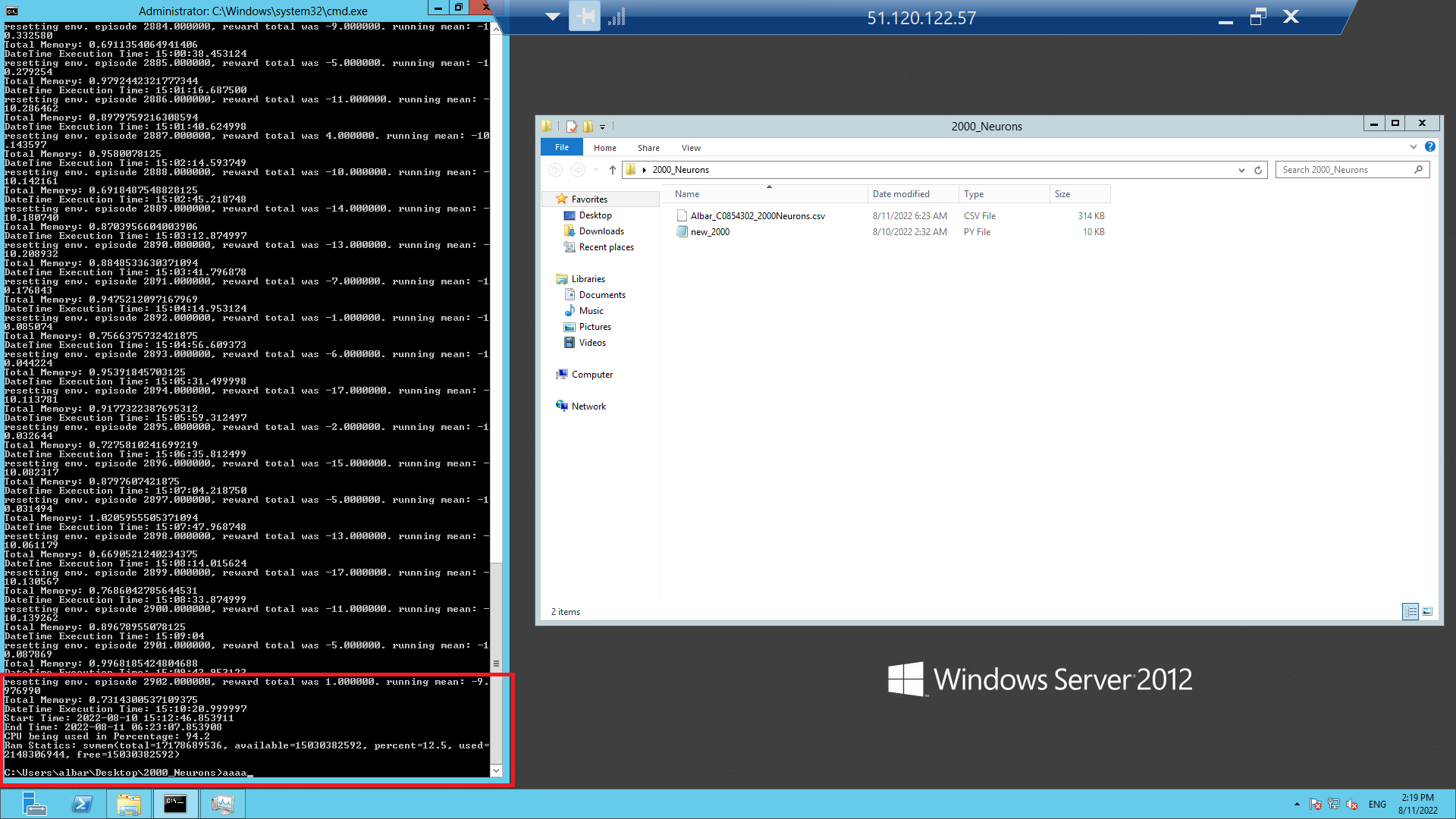
To execute the code for 2000 neurons and hit the running mean threshold of greater than -10, it took 15 hours 10 minutes, and 20.99 seconds, i.e., 54,620.99 seconds.



*Fig 9 – 2000 neurons Task Manager information\_1*



*Fig 10 – 2000 neurons Task Manager information\_2*

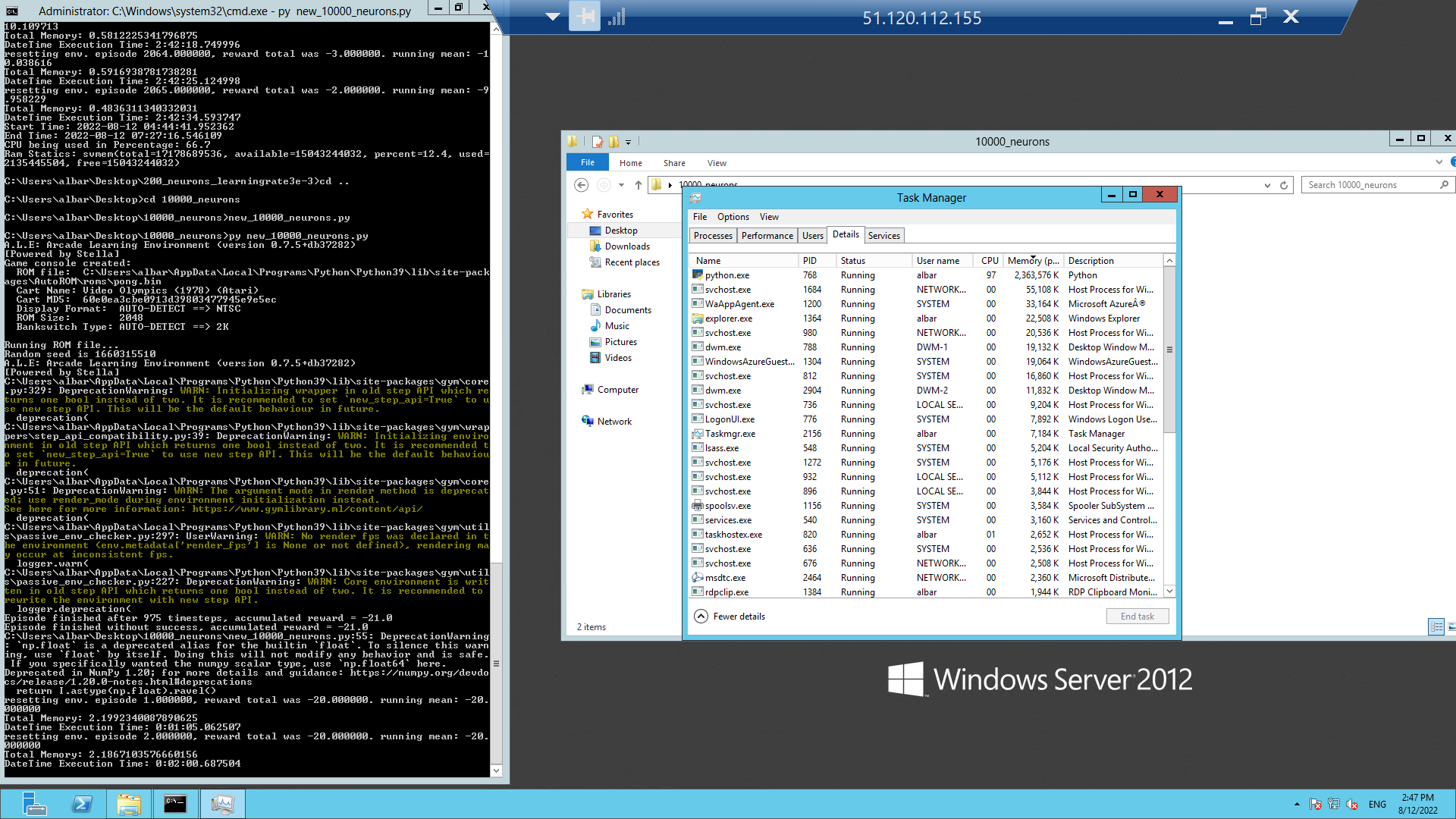


*Fig 11 – 2000 neurons termination screenshot*

10000 Neurons:

For 10000 neurons, unfortunately, I was unable to hit the threshold for either running mean greater than -10 or reaching 100,000 episodes due to Windows Server 2012 (virtual machine operating system) starting an update automatically.

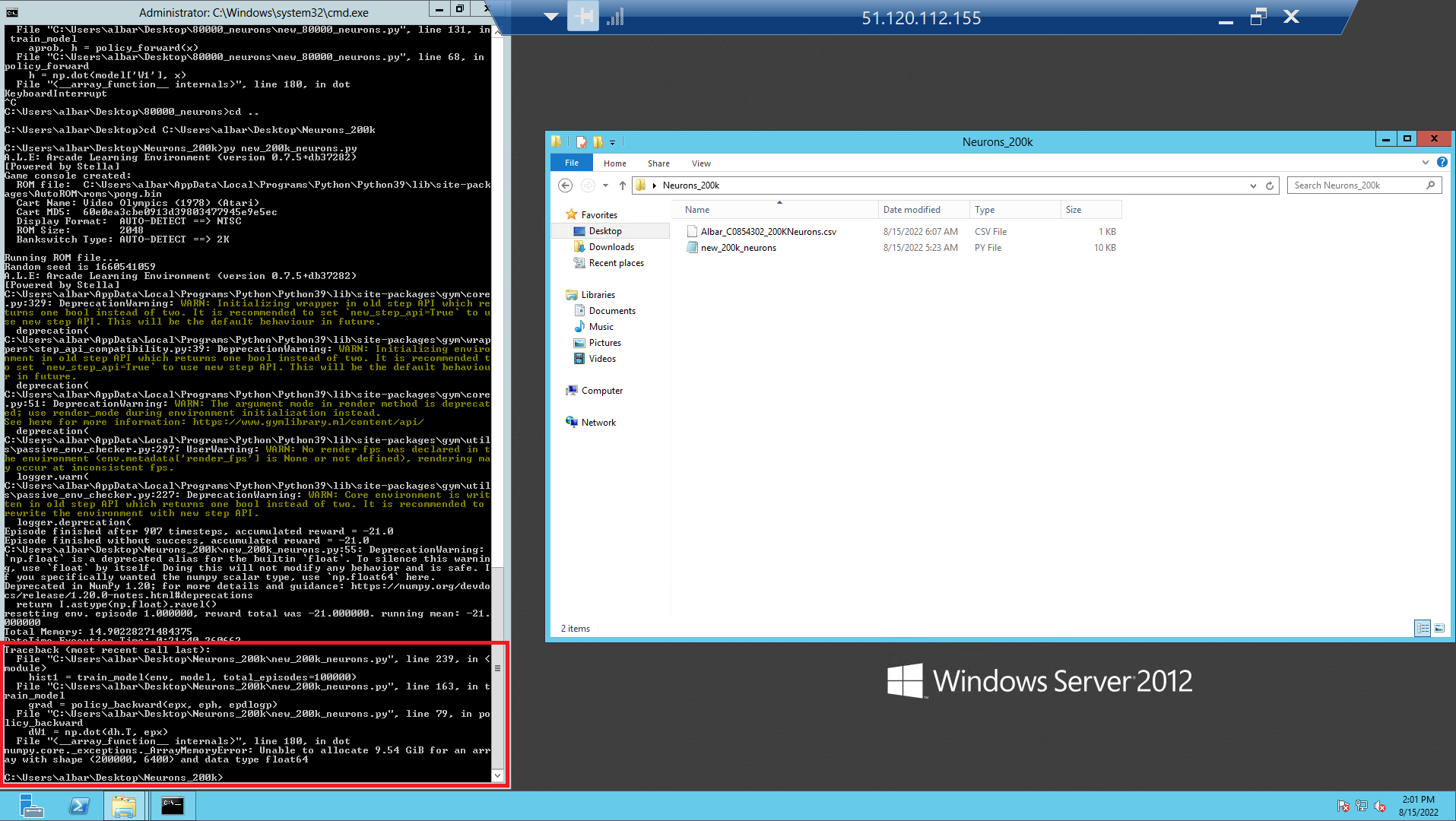
However, running mean of -11.61015998 was achieved in1493 episodes in an execution time of 47 hours, 16 minutes, and 49.09 seconds or 170,209.09 seconds. As per the forecast view, if the code ran for at least three more hours, it would have hit the threshold of running mean being greater than -10.



*Fig 12 – 10000 neurons screenshot (own photo)*

200,000 Neurons :

For 200,000 neurons, the code was executed for only one episode, and then the code crashed. This is due to the RAM and CPU both being overwhelmed due to a high number of neurons. It took 21 minutes and 40 seconds to execute just one episode.



*Fig 13 – 200,000 neurons crash*

**Graph of Time-Taken, RAM and CPU Used versus Number of Neurons:**

*Graph 1 – Neurons vs. Execution Time, RAM, and CPU*

With regards to learning rate, apart from the base learning rate i.e., 1e-04, I implemented 4 more which included **1e-05, 1e-03, 1e-02** and **3e-03,** with neuron value kept as constant, i.e., **200.** Out of these four learning rates, three hit the threshold of running mean and stopped, while one hit the episodes threshold and stopped.

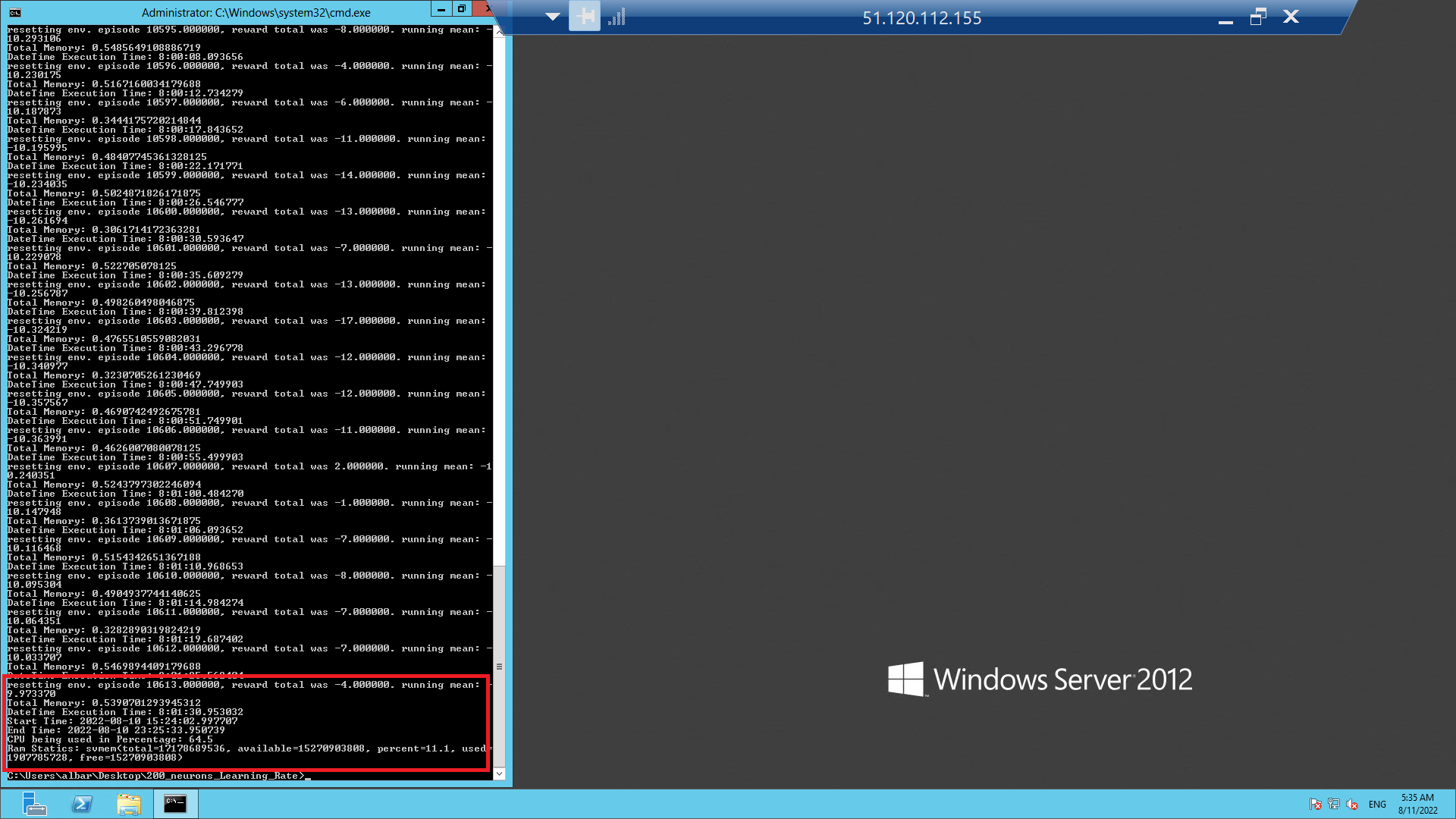
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test Iteration | Learning Rate | Start Time (time) | Stop Time (time) | Time Taken (seconds) | RAM Used | Episodes | Running Mean | CPU |
| 1 | 1e-04 | 10th Aug, 3:24:03 PM | 10th Aug, 11:25:34 PM | 28,891 seconds | 11.1% | 10613 | -9.973370052 | 64% |
| 2 | 1e-03 | 11th Aug,  5:44:46 AM | 11th Aug,  6:50:50 AM | 3,964 seconds | 13% | 1280 | -9.943645507 | 66.8% |
| 3 | 1e-02 | 11th Aug,  6:05:32 PM | 11th Aug,  3:38:29 AM | 34,377 seconds | 13.6% | 16806 | -9.948814605 | 67.8% |
| 4 | 3e-03 | 12th Aug,  4:44:42 AM | 12th Aug,  7:27:17 AM | 9,755 seconds | 12.4% | 2065 | -9.958229387 | 66.7% |
| 5 | 1e-05 | 11th Aug,  6:09:27 PM | 14th Aug,  8:13:12 AM | 223,424.83  Seconds | 11.4% | 100,000 | -14.40228609 | 69% |

***Table 2 – Learning based on Learning Rate***

**Learning Rate based Iterations –**

1e-04:

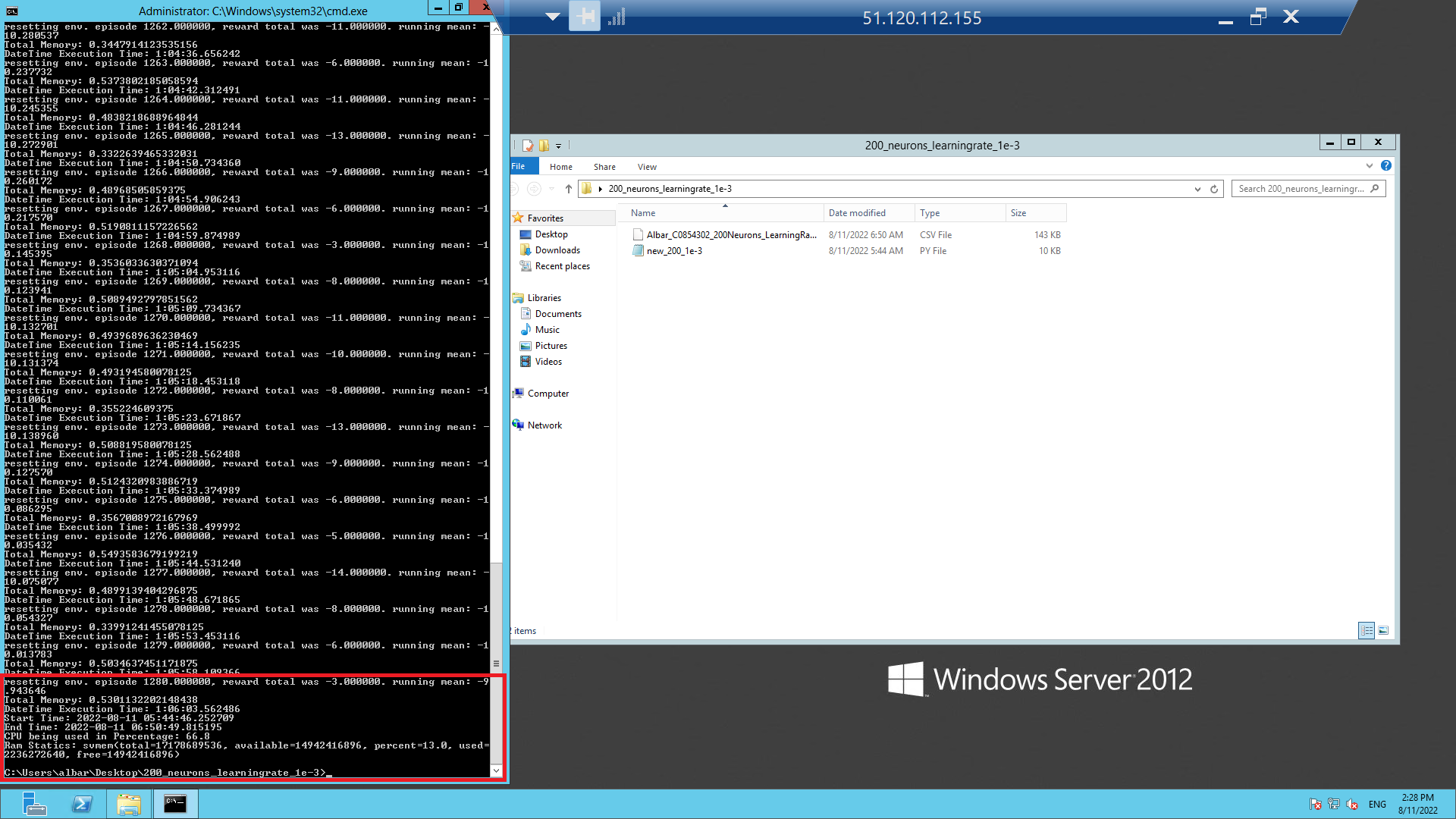
To execute the code for learning rate 1e-04 (the same as 200 neurons) and hit the running mean threshold of greater than -10, it took 8 hours 1 minute and 31 seconds i.e., 28,891.



*Fig 14 – Learning rate 1e-04 termination screenshot (own photo)*

1e-03:

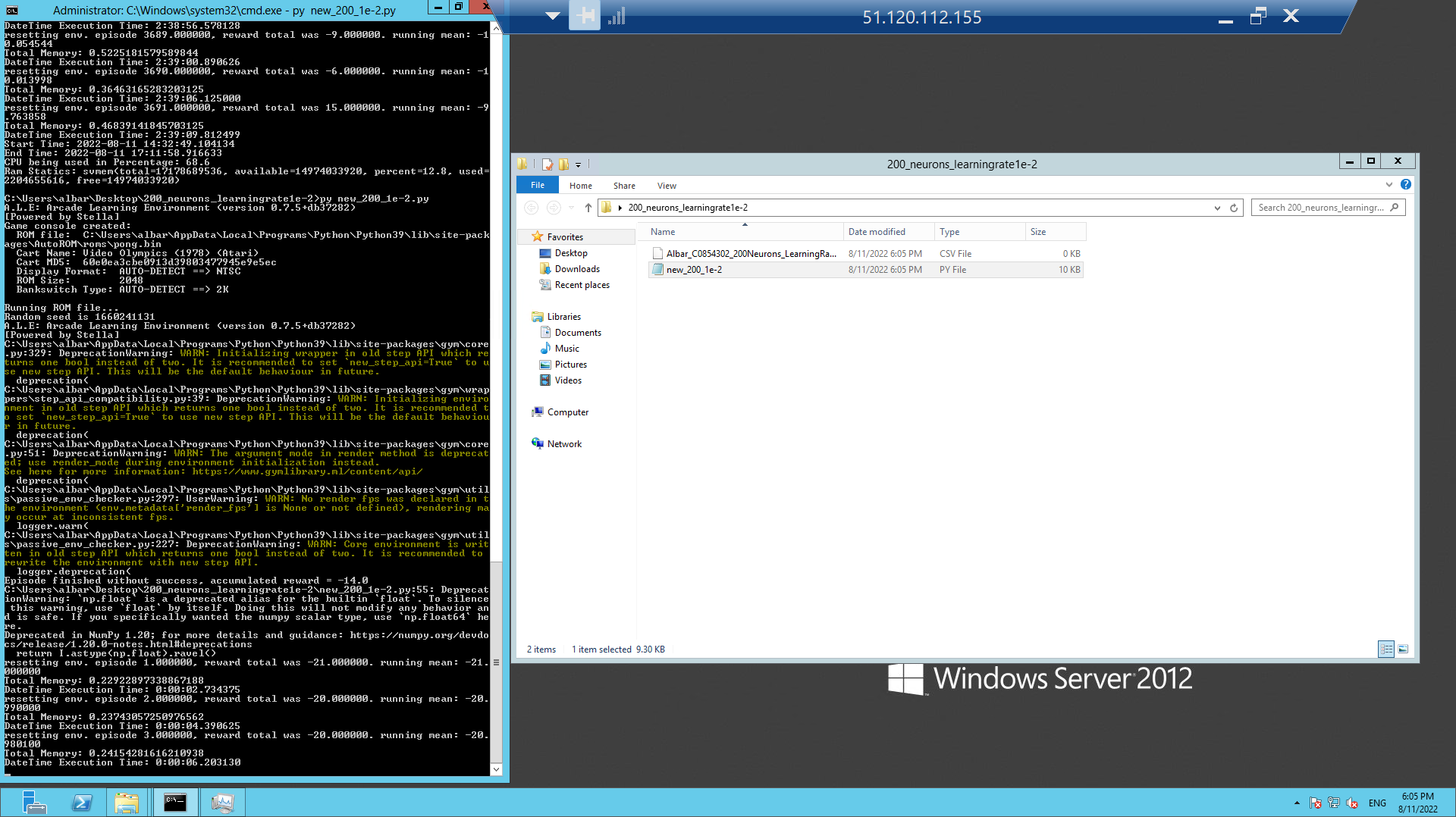
To execute the code for learning rate 1e-03 and hit the running mean threshold of greater than -10, it took 1 hour, 6 minutes, and 4 seconds, i.e., 3,964 seconds.



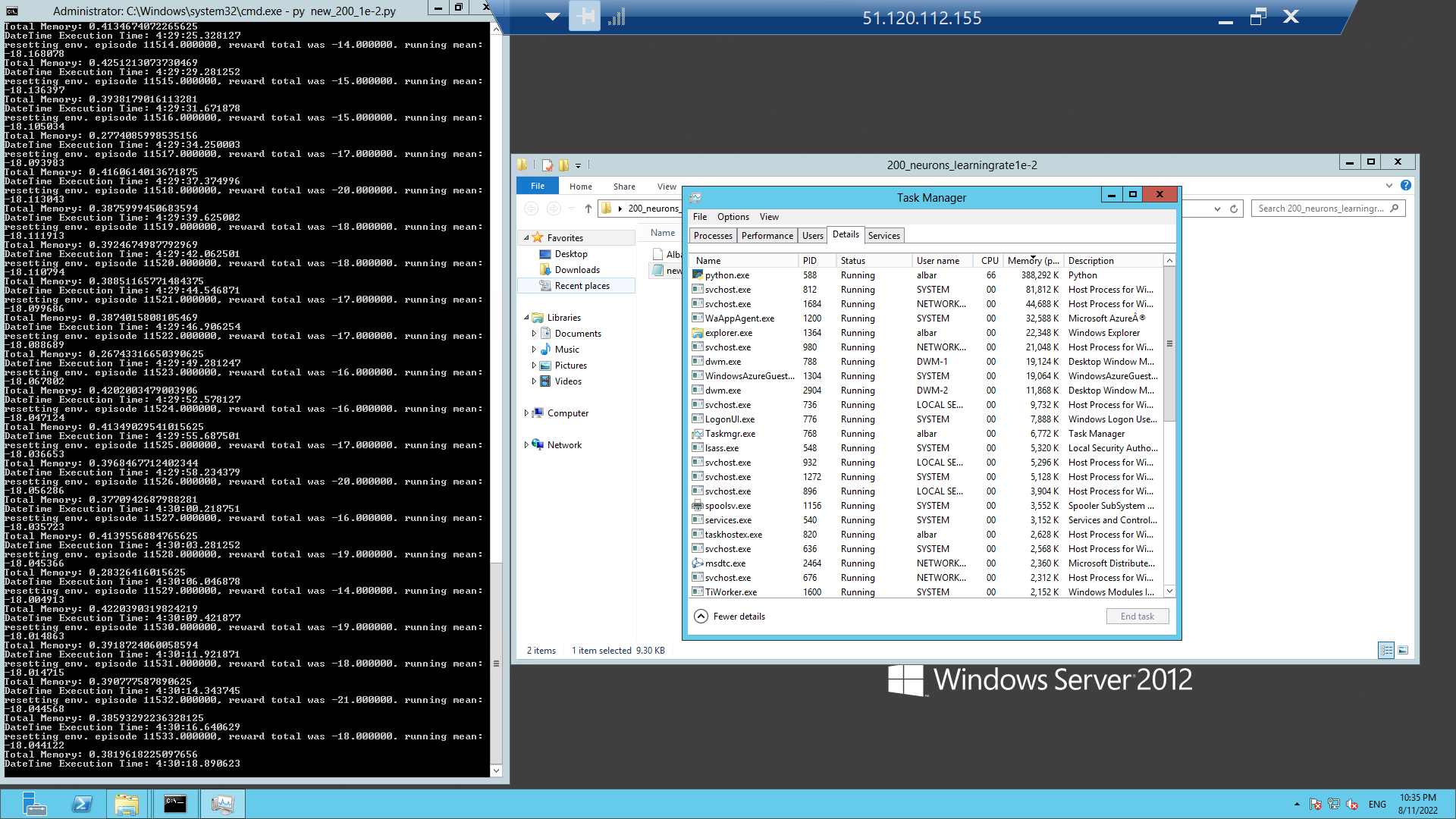
*Fig 15 – Learning rate 1e-03 termination screenshot (own photo)*

1e-02:

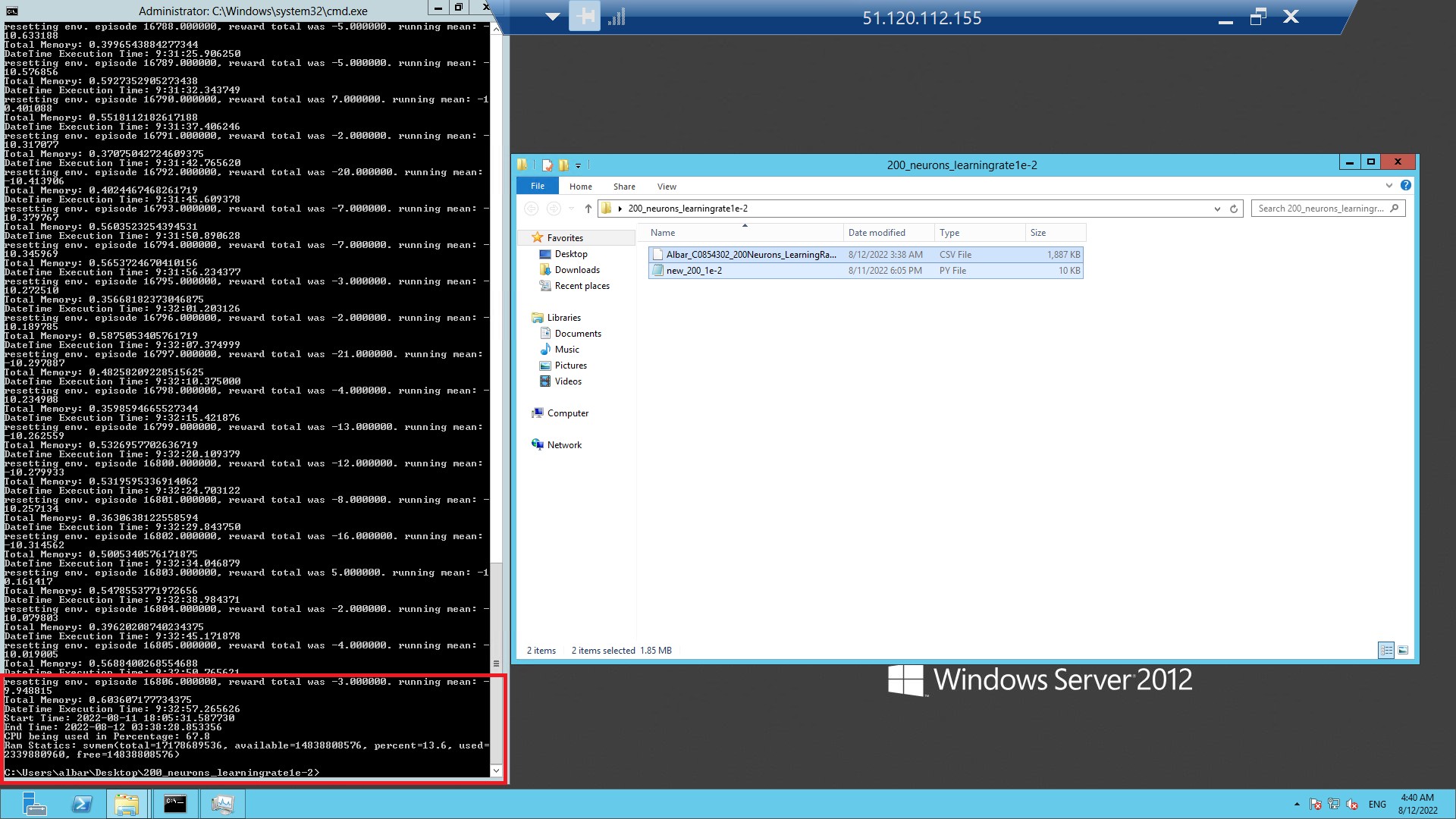
To execute the code for learning rate 1e-02 and hit the running mean threshold of greater than -10, it took 9 hours, 32 minutes, and 57 seconds, i.e., 34,377 seconds.



*Fig 16 – Learning rate 1e-02 Initiation*

**

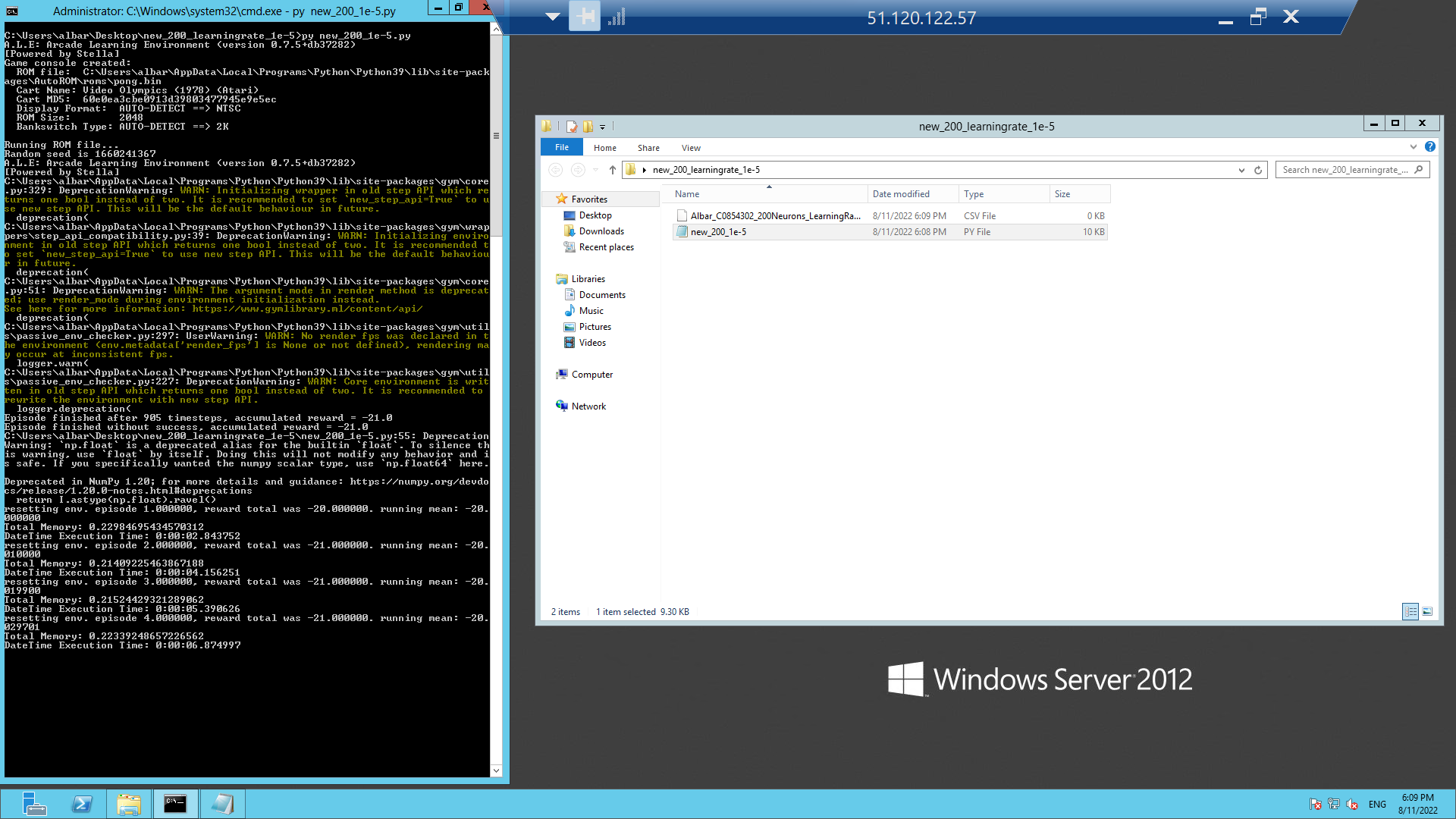
*Fig 17 – Learning rate 1e-02 mid-way with Task Manager*

**

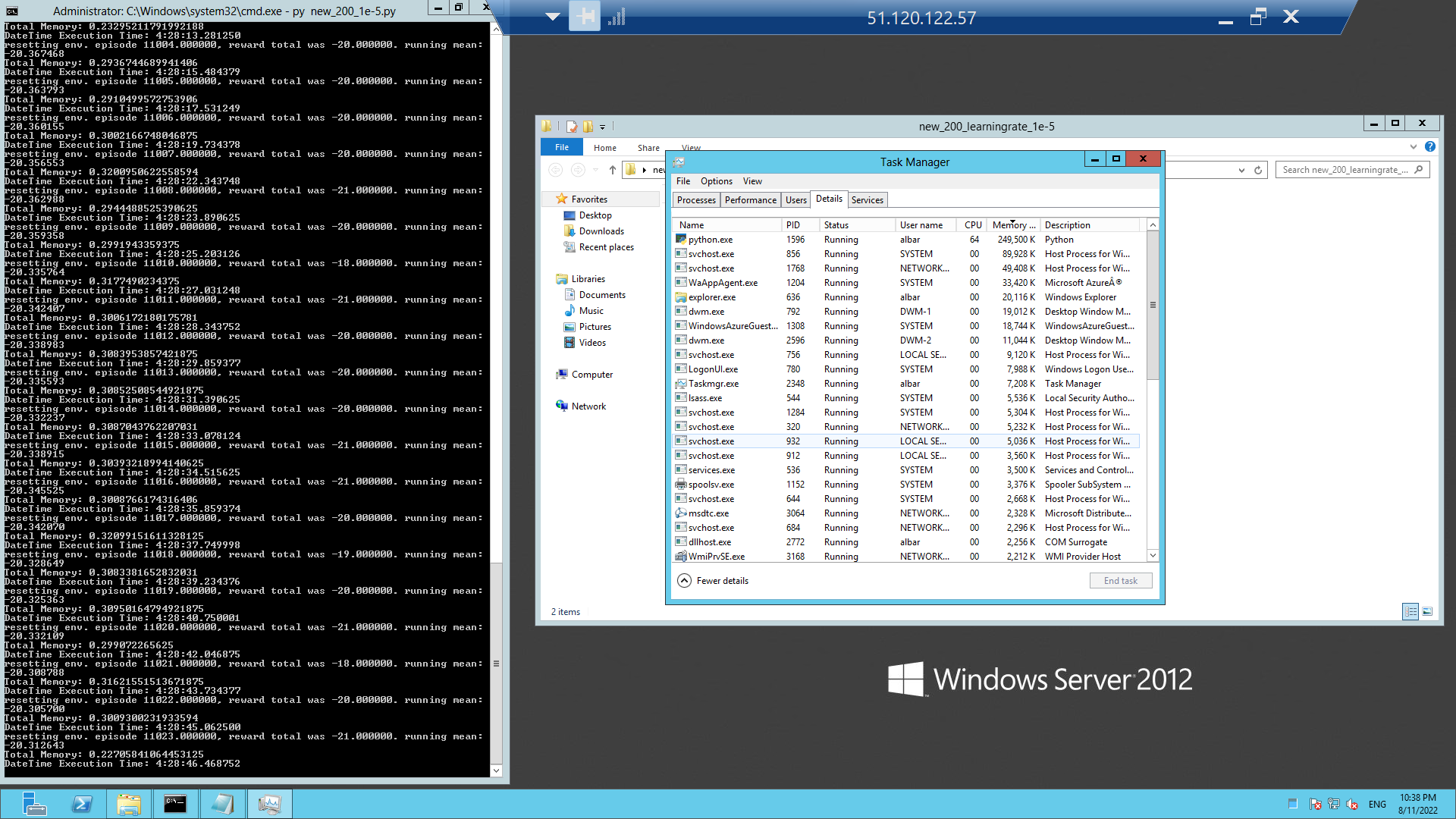
*Fig 18 – Learning rate 1e-02 termination screenshot*

1e-05:

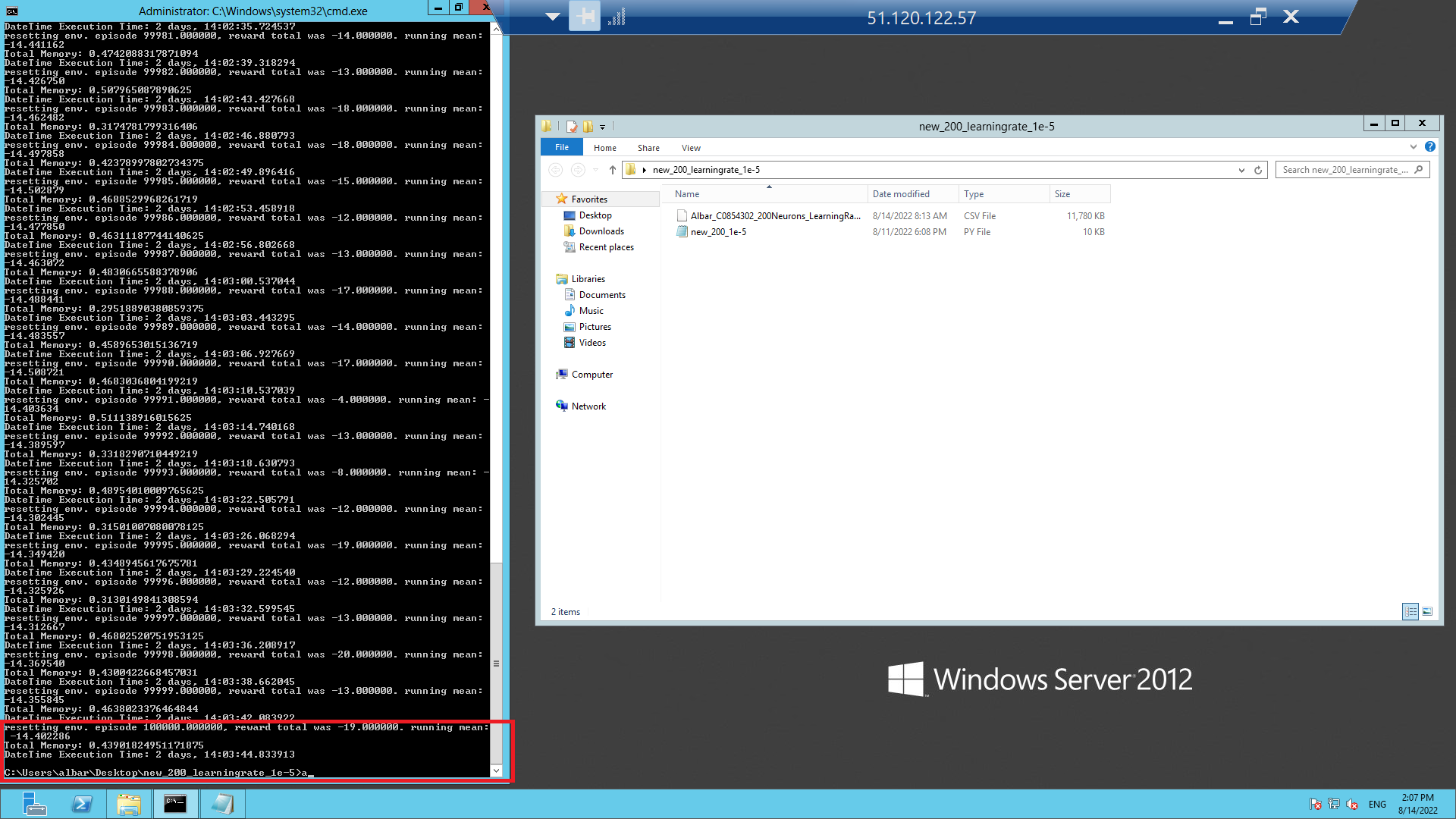
For learning rate 1e-05, instead of hitting the running mean threshold of greater than -10, the episodes 100,000 thresholds were hit. It took 62 hours 03 minutes and 44.83 seconds, i.e., 223,424.83 seconds.



*Fig 19 – Learning 1e-05 initiation screenshot*

**

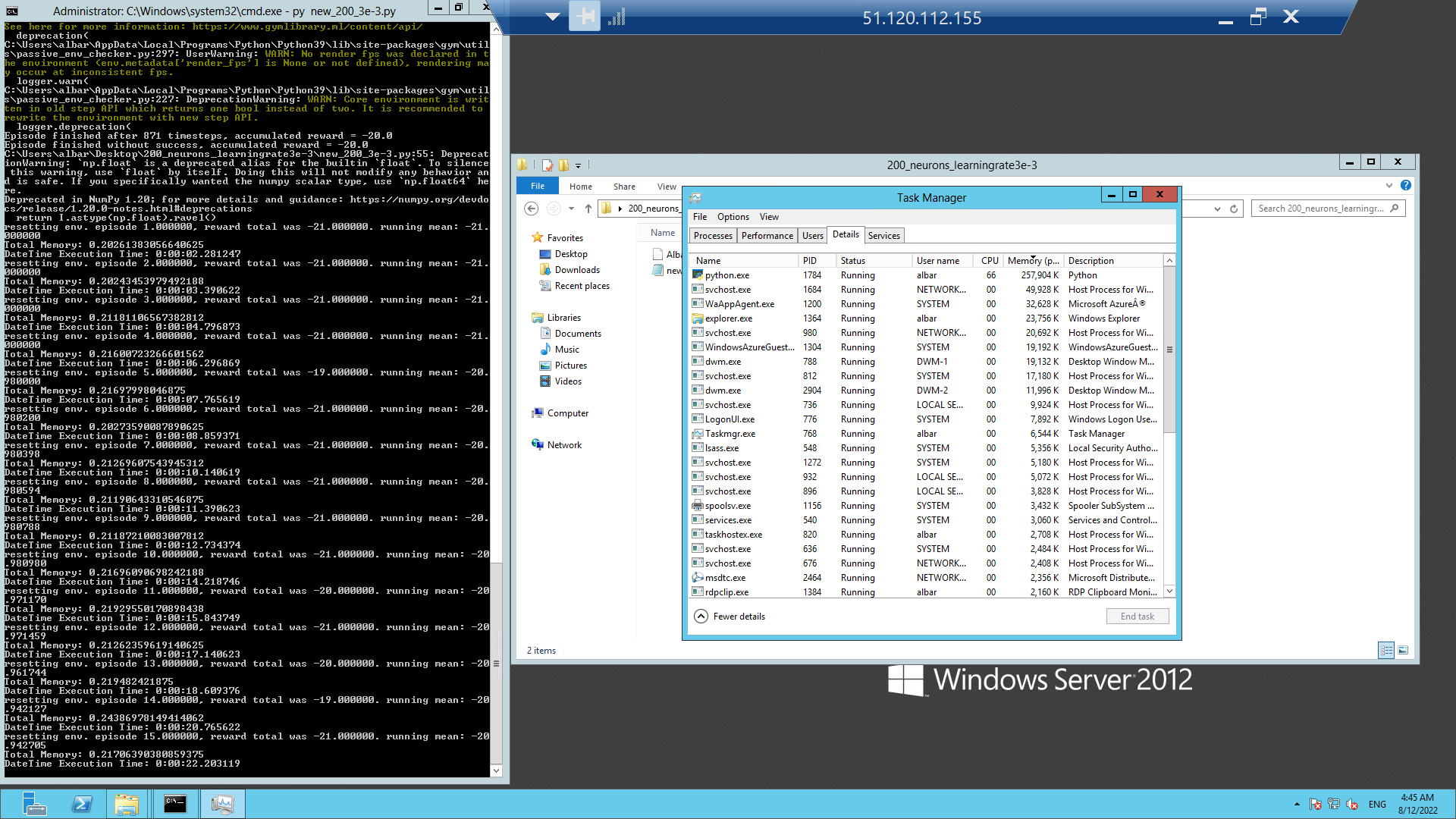
*Fig 20 – Learning 1e-05 mid-way with task manager*

**

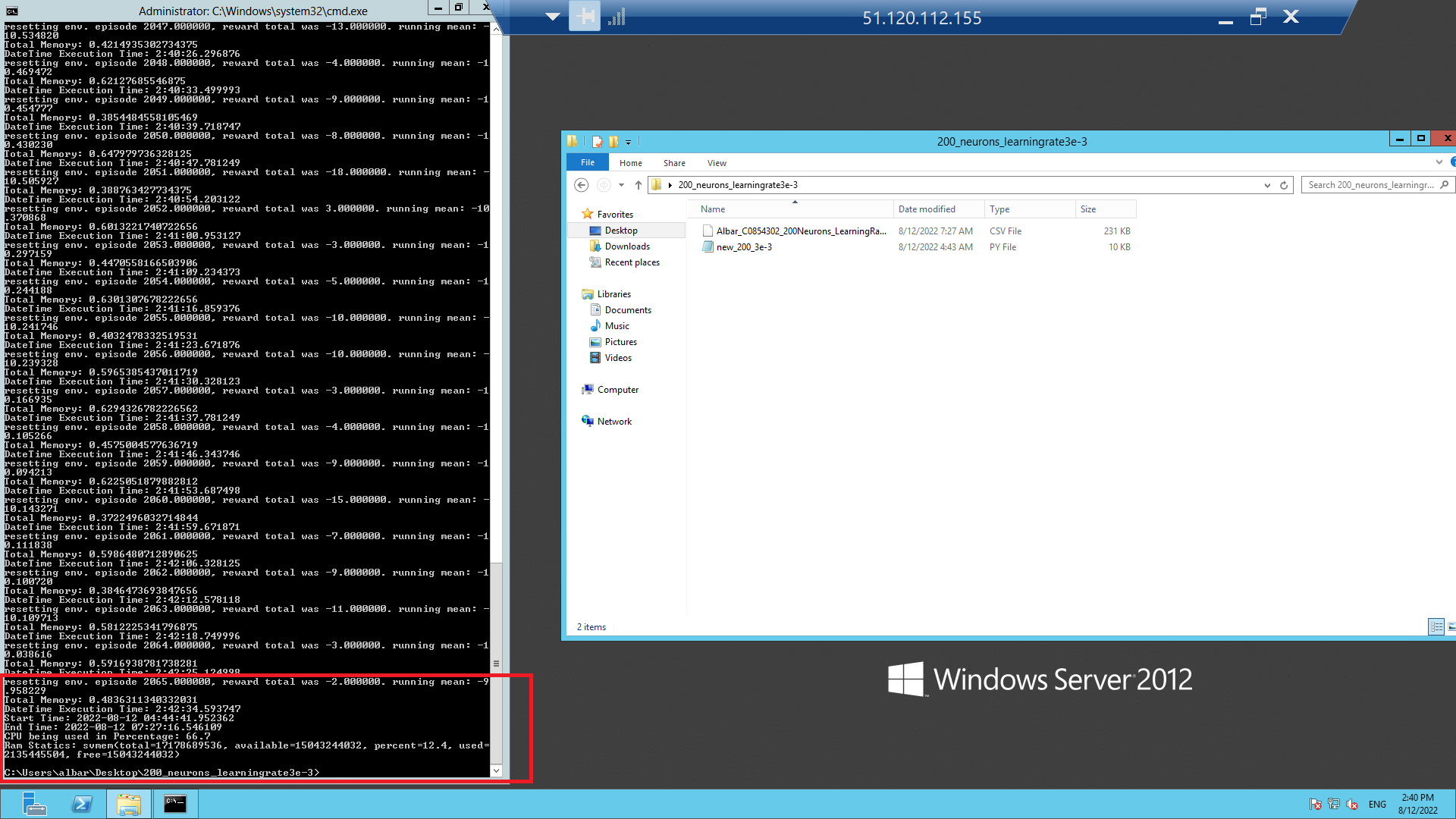
*Fig 21 – Learning 1e-05 termination screenshot*

3e-03:

To execute the code for learning rate 3e-03 and hit the running mean threshold of greater than -10, it took 2 hours 42 minutes and 35 seconds, i.e., 9,755 seconds.



*Fig 22 – Learning 3e-03 initiation with task manager*



*Fig 23 – Learning 3e-03 termination screenshot*

**Graph of Time-Taken, RAM and CPU Used versus Learning Rate:**

*Graph 2 – Learning Rate vs. Execution, RAM, and CPU*

**Conclusion**

Considering the analysis done on an azure machine with 2vCPUs and 16 GIB RAM, with respect to neurons, it is safe to say that the higher the number of neurons, the faster the learning, i.e., a smaller number of episodes required to achieve the threshold, however, time taken to complete each episode increased significantly (the time taken to complete each episode can be variable, as it would depend upon the number of CPUs, its speed and processing time)and also the ram or memory used to complete each episode increased as well [Table 1]. Since 10,000 neurons did not meet the threshold as required due to automatic operating system updates and 200,000 neurons code crashed, it can be said that 2000 neurons achieved the best learning.

With respect to learning rate, it is similar to neurons. The higher the learning, the better and faster the result. The difference that could be seen is that since the neurons were constant, a higher learning rate did not require a higher amount of time to complete an individual episode, and the CPU and RAM utilized remained pretty much in the same bracket with only a slight increase or decrease [Table 2]. When the learning rate was lowered, the number of episodes and time required to converge the running mean significantly increased, and this phenomenon can be seen with learning rate 1e-5, where the code stopped after hitting the 100,000-episode threshold as opposed to the -10 running mean threshold. Lastly, it can be seen that learning rate 1e-3 required the least number of episodes as well as time to hit the running mean threshold of -10.

**References**

1 - <https://gist.github.com/karpathy/a4166c7fe253700972fcbc77e4ea32c5>

2 - <http://karpathy.github.io/2016/05/31/rl/>

3 -<https://colab.research.google.com/drive/1KZeGjxS7OUHKotsuyoT0DtxVzKaUIq4B?usp=sharing>