Do you want to convert your existing Laptop into an AI Laptop?

# Purpose and Background

We define an AI laptop as one with an AI-capable processor. This write-up shares our experience **of converting our laptop into an AI laptop by running the Ollama model with 7B parameters on our regular laptop**.

The quantized Open Large Language Model Meta AI(LLaMA) is developed by Meta AI. Unlike other cloud-based models like ChatGPT, LLaMA can be run locally, giving the benefits of cost-savings, privacy, offline access, GPU acceleration and CLI-based experience.



# System Requirements

For the standard system requirements for Ollama installation, refer to <https://ollama.com/> . Our system configuration is

* OS: Windows 11 Home Single Language
* CPU: x86\_64 architecture
* GPU: NVIDIA GeForce RTX 3070
* CPU RAM: 32 GB
* GPU: global memory: 8192 MBytes

## Note

* Before you start converting your laptop, ensure you have a supported GPU installed on your system. Refer to the support matrix **https://docs.nvidia.com/deeplearning/cudnn/backend/latest/reference/support-matrix.html**
* We have explained how to run LLaMA2 – 7B. Here 7B refers to 7 billion parameters. Our choice of 7B is dictated by CPU RAM(32GB) and GPU global memory(8GB) capacities. If we had more memory, we could have run bigger sized models.

# CUDA Toolkit

We wanted LLaMA to make use of our GPU for accelerating the model inference performance. To achieve this, we first installed correct version of CUDA on a compatible NVIDIA GPU available on our system. In our case it is **CUDA Toolkit 12.9 Update 1.** For the Standard system requirements and instructions for CUDA installation, refer to [**CUDA Toolkit 12.9 Update 1 Downloads | NVIDIA Developer**](https://developer.nvidia.com/cuda-downloads?target_os=Windows&target_arch=x86_64&target_version=11&target_type=exe_local)

# Building our AI platform

## Implementation Steps

Implement the LLaMA in five-steps.

1. Install CUDA
2. Validate installation
3. Download and run Ollama
4. Validate - Is the model really leveraging GPU?
5. Inference using LLaMA

### 1.Installing CUDA Toolkit 11.8+

In the previously mentioned NVIDIA download link, choose an image as per following specifications.

* Operating System: Windows
* Architecture: x86\_64
* Version: Windows 11
* Installer Type: exe (local)

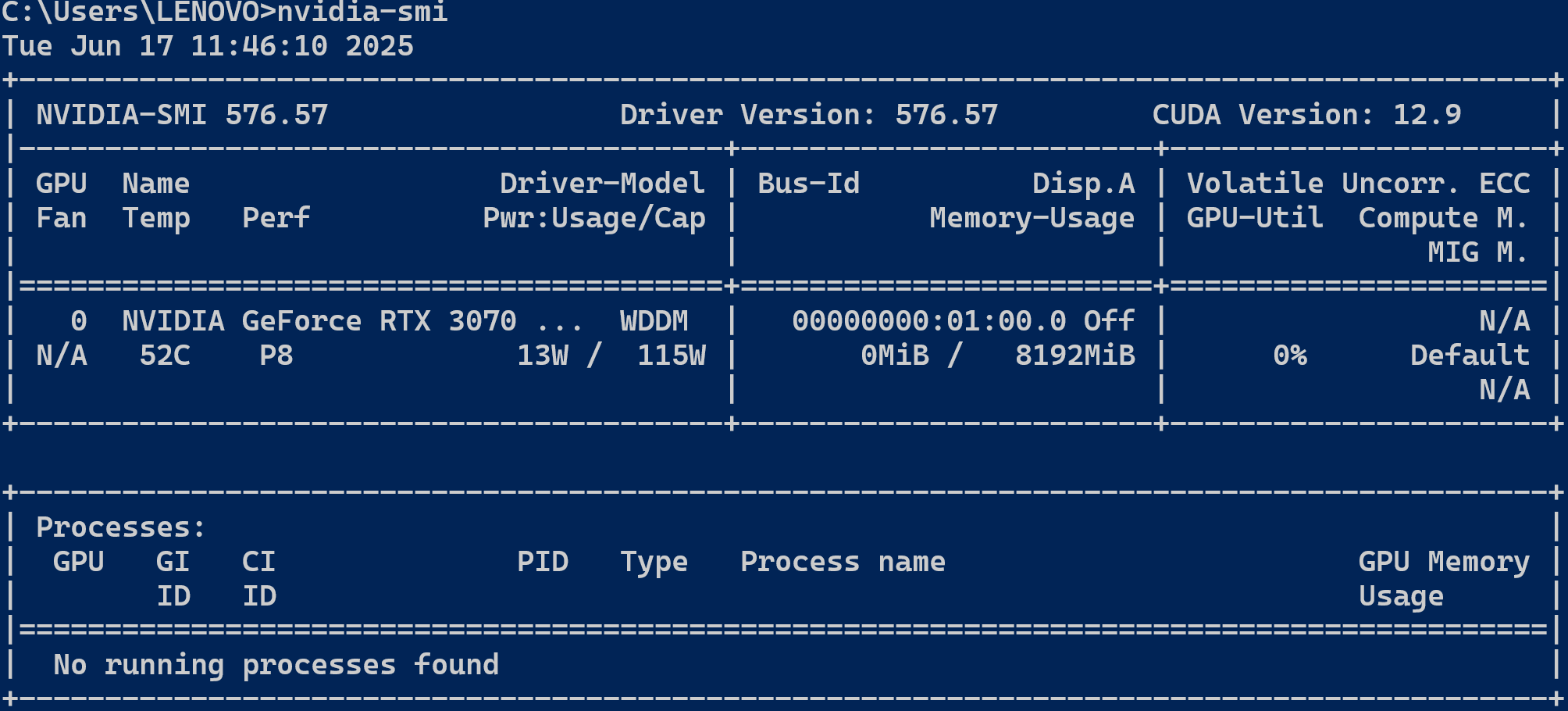
Further clicking the Download button copies thecuda\_12.9.1\_576.57\_windows.exe, 3.3 GB in size into a local folder.Finally,click the Windows file to install CUDA.

### 2.Validating the installation

Verify the CUDA installation by typing

**nvidia-smi**

Following visual provides all information regarding GPU and CUDA. Importantly the CUDA version is 12.9.



In the location **cd "C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v12.5/extras/demo\_suite"** run the command **deviceQuery.exe** to get additional GPU details.

### 3.Downloading and running the Ollama

Ollama is a tool that lets users download and run models like LLaMA, Mistral, Gemma, etc. locally on their device. Ollama provides a command-line interface and a local HTTP API so that users can chat with models or integrate them into apps — all without internet or API keys. It’s optimized for tasks like text generation, question answering, and summarization.

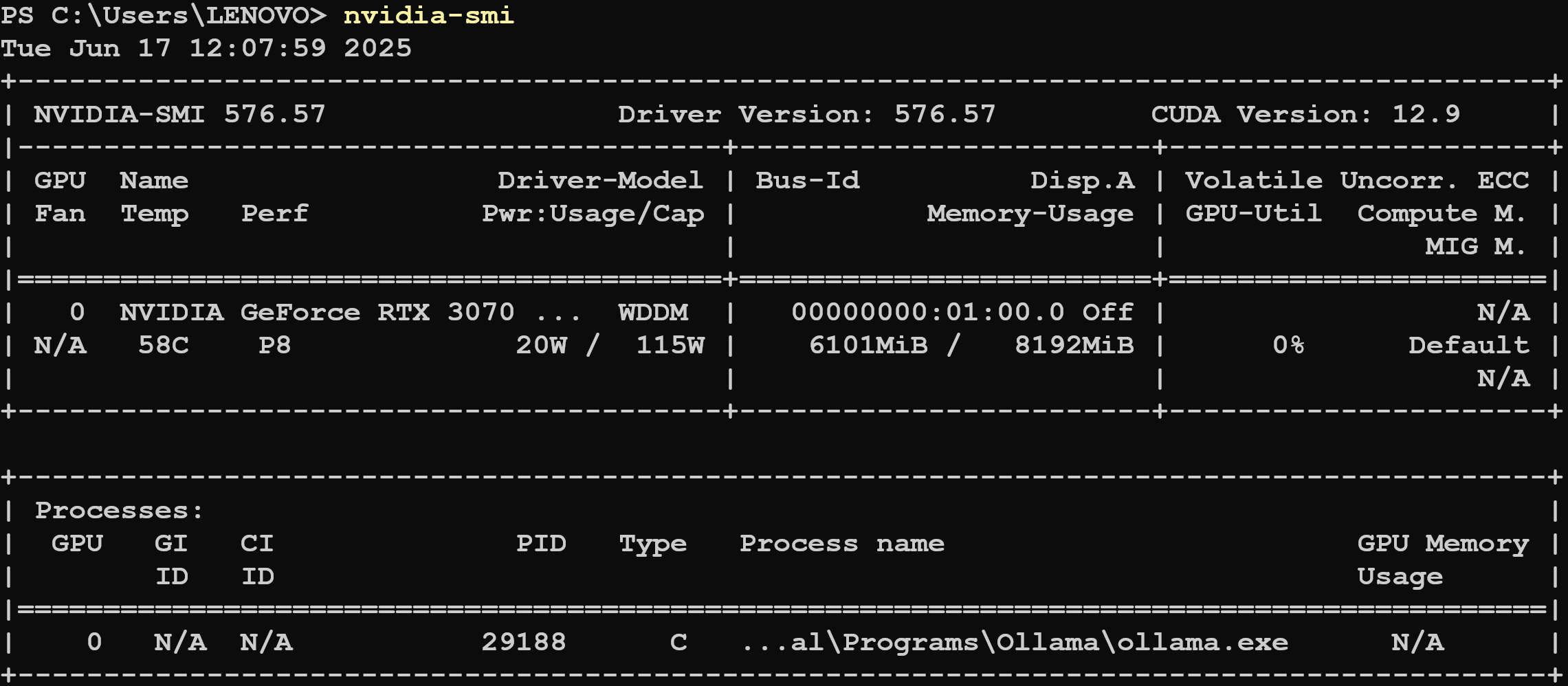
From [**https://ollama.com/download**](https://ollama.com/download) , get the Ollama file (3.8 GB) and run. The executable gets installed in C:\Users\LENOVO\AppData\Local\Programs\Ollama. This location is called OLLAMA\_ROOT

### 4.Is the model really utilizing GPU ?

We first launched LLaMA using

**Ollama run llama2**

to ensure that the model is leveraging the GPU. Then we launched **nvidi-smi** again.



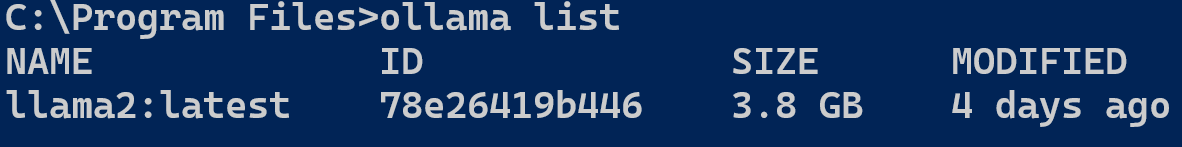
This output confirms that Ollama.exe is using the GPU and it consumes 6101 MiB GPU global memory. Recall when it was not running on GPU, processes entry was empty and GPU memory used was 0. This is a validation that the model is loaded and is running on the GPU.

### 5.Inferencing using LLaMA with sample chat

Open a PowerShell and go tothe locationOLLAMA\_ROOT The correct way is to set the path for this folder. Before inferencing,first check the version of LLaMA by typing

**ollama list**

and the output produced is



To run the model, type

**Ollama run llama2**

*pulling 8934d96d3f08: 100% ▏ 3.8 GB/3.8 GB 12 MB/s 0s*

*pulling 8c17c2ebb0ea: 100% ▕▏ 7.0 KB*

*pulling 7c23fb36d801: 100% ▕▏ 4.8 KB*

*pulling 2e0493f67d0c: 100% ▕ 59 B*

*pulling fa304d675061: 100% ▕▏ 91 B*

*verifying sha256 digest*

*writing manifest*

***success***

### Sample chats

**>>> Write a Python function to check if a number is prime?**

Prompt: Write a Python function to check if a number is prime

Response Time: 5.338338 seconds

Response:

Here is an example of a Python function that checks whether a given number is prime:

```

def is\_prime(n):

"""Return True if the number is prime, False otherwise"""

if n <= 1 or n % 2 == 0:

return False

for i in range(2, int(n \*\* 0.5) + 1):

if n % i == 0:

return False

return True

```

Explanation:

\* The function takes a single argument `n`, which is the number to be checked for primality.

\* The function first checks whether `n` is less than or equal to 1, in which case it is always composite (not prime).

\* If `n` is greater than 1, the function then checks whether it is divisible by any integer between 2 and `n \*\* 0.5`. If it is divisible by any of these numbers, it is not prime.

\* Otherwise, the function returns True, indicating that the number is prime.

Here is an example of how to use the function:

```

>>> is\_prime(10)

True

>>> is\_prime(8)

False

>>> is\_prime(4)

True

```

Note that this function has a time complexity that its running time grows rapidly as the input number increases. For large numbers, other primality tests with lower time complexities may be more appropriate.

### Response Time

Using the following script, we measured the performance of the model running on AI laptop.

# ollama-timer.ps1

$start = Get-Date

$response = ollama run llama2 "Write a Python function to check if a number is prime"

$end = Get-Date

$duration = $end - $start

Write-Host "`n Prompt: Write a Python function to check if a number is prime"

Write-Host " Response Time: $($duration.TotalSeconds) seconds"

Write-Host "Response:"

$response

|  |  |  |
| --- | --- | --- |
| Test | Avg. Response time in secs. | Remarks |
| Repetition test | 8.61928252 | 5 runs |
| Code generation | 5.59563806 | 5 runs |

# Key Takeaways

1. Convert your existing Windows laptop into an AI laptop. Use this AI laptop for question-answering and summarization in a secure way with minimal cost
2. Benefit from other models like LLaMA 3.3, DeepSeek-R1, Phi-4, Gemma 3, Mistral Small 3.1 and others.