# Introduction

The **free\_volume.py** module of **LUNAR** supports CUDA run modes via the numba python package. Where the tasking part of the code will transfer data to a GPU and perform operations on that data using a GPU kernel that runs in parallel and then transfer the results back to the CPU. The unfortunate part of this run mode however is that you must have an NVIDIA GPU and the corresponding CUDA drivers. The method explained in this document to install CUDA drivers will allow you to run the CUDA modes regardless of how you have installed Python on your machine (i.e. through <https://www.python.org/> or <https://www.anaconda.com/> or command line in Linux). The tutorial will be for installing CUDA drivers for Windows, but installing CUDA drivers for Linux will follow the same basic steps.

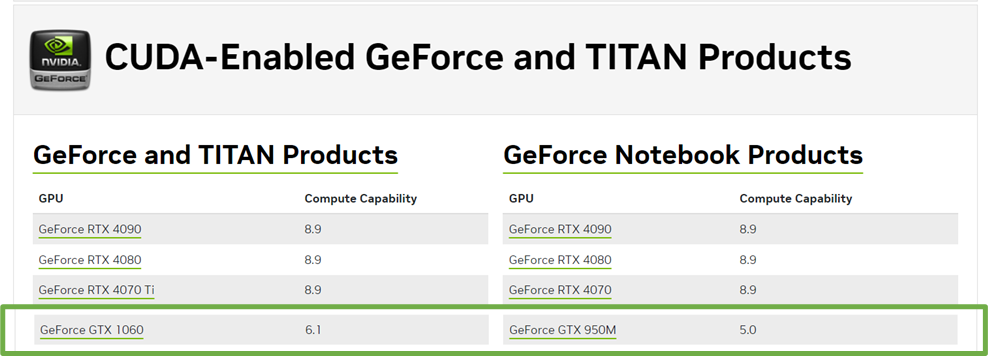
One note is that some users may find the following steps to much and just decided to skip setting up CUDA run mode support. This is entirely their decision, where multiple factors may play into the decision, such as limited storage on their OS drive since both NVIDIA GeForce Experience and VS are required (both of which take up a fair amount of space). A few perspectives would like to be given hear to guide the decision, such as most common place CPUs have less than 16 cores and typically a GPU have several thousand cores. This means for large systems (>10,000 atoms) with small voxel sizes (<0.2 Å) the GPU will start to seriously outperform the CPU, since it can run a lot more tasks concurrently. The offloading of tasks from the CPU to the GPU also allows the users PC to be used during a free volume calculation, since the CPU will only be used for I/O operations and not a larger number of mathematical operations. Finally, concerns about increasing the temperature of the CPU above 80 °C for longer periods of time and the possible damage that may occur. The heat generation of the CPU will be dependent on your machine; however, GPUs typically run much cooler and will run more task concurrently avoiding large amounts of heat generation preserving your machine.

# Setup

1. Open a Windows command prompt (cmd) or Linux terminal and type “nvidia-smi --list-gpus”, which should create an output like the one below (note if you have a “command center” that was distributed with your PC, you can also find out your GPU model that way as well).



1. Open the <https://developer.nvidia.com/cuda-gpus> link to see if your GPU supports CUDA.



1. If your GPU supports CUDA get the drivers from this link <https://developer.nvidia.com/cuda-downloads>.

A screenshot of a computer

Description automatically generated

1. Before installing CUDA drivers you will need to install the community Visual Studio (VS) from this link <https://visualstudio.microsoft.com/> (or if on Windows from the Microsoft store). Follow the prompts of the installation GUI.

A screenshot of a computer

Description automatically generated

1. Install the CUDA drivers via the CUDA installation GUI.

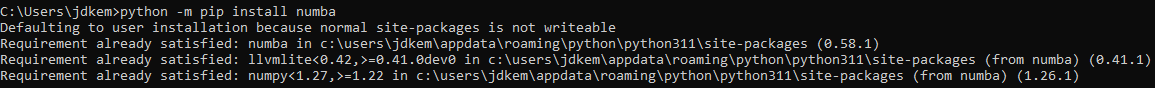
A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Install the tqdm, numpy, and numba Python packages by running:
   1. tqdm
      1. pip install tqdm (for Anaconda and Linux Python installs)
      2. python -m pip install tqdm (for Python install from <https://www.python.org/>)
   2. numpy
      1. pip install numpy (for Anaconda and Linux Python installs)
      2. python -m pip install numpy (for Python install from <https://www.python.org/>)
   3. numba
      1. pip install numba (for Anaconda and Linux Python installs)
      2. python -m pip install numba (for Python install from <https://www.python.org/>)
   4. If already installed you’ll see messages like such, or it will install the package



1. Run **LUNAR.py** from IDE or command line or cmd to get access to the master LUNAR GUI and click on the **free\_volume** button.

A screenshot of a computer

Description automatically generated

1. Set the **run\_mode** to CUDA, the **CUDA\_threads\_per\_block\_atoms** to 8, and the **CUDA\_threads\_per\_block\_voxels** to 16. Then click “Run LUNAR/free\_volume.py” button and you will be running your first parallel CUDA calculation. Where you will see (CUDA …) showing which parts of the code are being offloaded onto the GPU.

A screenshot of a computer

Description automatically generated

1. Please see **LUNAR\_manual\_v1.doc** for more details about LUNAR and the different features of **free\_volume.py** and how to choose appropriate settings for the CUDA run modes.