How to use local GPU in Jupyter? (Windows 10)

- Ensure your device has GPU drivers installed. Let it be the NVIDIA RTX A4000.
- Check its compute capability here. Its 8.6. Tensorflow supports 3.5 and higher.
- Install the latest version of Anaconda Navigator (Google it!)
- Launch CMD.exe Prompt application from the Navigator
- Check the python version by typing 'python --version'. Say it's 3.10.9.
- Check the versions of tensorflow (GPU), CUDA, cuDNN compatible with this python version here. For python 3.10, the latest ones are 2.10.0, 11.2, 8.1 respectively.
- Install Microsoft Visual Studio and Visual C++ Redistributable
- Install the desired version of CUDA (11.2 here) from CUDA Toolkit Archive | NVIDIA
- Follow the instructions to install cuDNN on your OS: Installation Guide :: NVIDIA
- Create and activate a new environment by typing the following in the prompt:

  `conda create -n gpu`

  `conda activate gpu` (You can use any name of your choice inplace of gpu)

  See Managing environments — conda documentation for details

- Install python kernel in the gpu environment to be used by jupyter. Type:

  ```
  pip install ipykernel
  python -m ipykernel install --user --name gpu
  pip install jupyter notebook
  ```

- Next Install TensorFlow with pip by typing in the following within the prompt:

  (I think this statement may not be required if CUDA+cuDNN was manually installed)
  `conda install -c conda-forge cudatoolkit=11.2 cudnn=8.1.0`

  (Anything above 2.10 is not supported on the GPU on Windows Native)
  `python -m pip install "tensorflow<2.11"`

  (Verify the installed tensorflow is able to detect a GPU device)
  `python -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"`

- In the home tab of Navigator, switch to gpu (from base) next to All applications on
- Install/launch the Jupyter Notebook application and create/open the desired notebook ensuring the kernel chosen is gpu.

- Type the following in the notebook to verify if the Jupyter is GPU-enabled:

  ```
  import tensorflow as tf
  ```

  ```
  tf.__version__
  ```

  Checks the version of the installed tensorflow

  ```
  tf.test.is_built_with_cuda()
  ```

  Checks if the installed tensorflow supports CUDA

  ```
  from tensorflow.python.client import device_lib
  ```

  ```
  device_lib.list_local_devices()
  ```

  Checks all local computing devices accessible to tensorflow

  ```
  tf.config.list_physical_devices('GPU')
  ```

  Checks all GPU devices accessible by tensorflow

  ```
  tf.test.gpu_device_name()
  ```

  Returns the available device in use by tensorflow currently

  You can now run your code. All possible GPU computations will use the GPU.

- See Enable TensorFlow-gpu with NVIDIA graphics on Windows 10 | by Koushik kumar | Analytics Vidhya | Medium for a detailed reference

Follow a similar procedure for utilising GPU in Pytorch (only new/additional steps):

- Launch CMD.exe Prompt application from the Navigator
- Pytorch only supports Python 3.7-3.9. So create a new environment with 3.9:
  `conda create -n ptgpu python=3.9 anaconda` (Use any name in place of ptgpu)
- Activate this environment by launching CMD.exe from ptgpu in Navigator home
- Execute the following in this command prompt:

  ```
  pip install ipykernel
  ```

  ```
  python -m ipykernel install --user --name ptgpu
  ```

  ```
  pip install jupyter notebook
  ```

  ```
  conda install -c conda-forge cudatoolkit=11.2 cudnn=8.1.0
  ```

- Find compatible Pytorch installation (Note: explicit version for cuda 11.2 doesn't exist)

  The below continuous code worked on my system:

```
pip install torch==1.9.1+cu111 torchvision==0.10.1+cu111
torchaudio==0.9.1 -f
https://download.pytorch.org/whl/torch_stable.html
```

(Verify the installed torch can access cuda)

```
python -c "import torch; print(torch.cuda.is_available())"
```

- Install/launch the Jupyter Notebook application and create/open the desired notebook ensuring the kernel chosen is ptgpu
- Type the following in the notebook to verify if the Jupyter is GPU-enabled:

  ```
  import torch
  torch.cuda.is_available()
  ```

  Checks if the installed torch supports CUDA

  ```
  torch.cuda.get_device_name(0)
  ```

  Returns the first available GPU device