17 Minutes or 7 Hours? Train ML Model Using GPU vs CPU

https://github.com/tuneman7/cuda_work/blob/main/README.md

Prepared For:
- *Data Science and Machine Learning Students and Practitioners*

Prepared By:
Don Irwin

W255 Summer 2022
August 6 2022

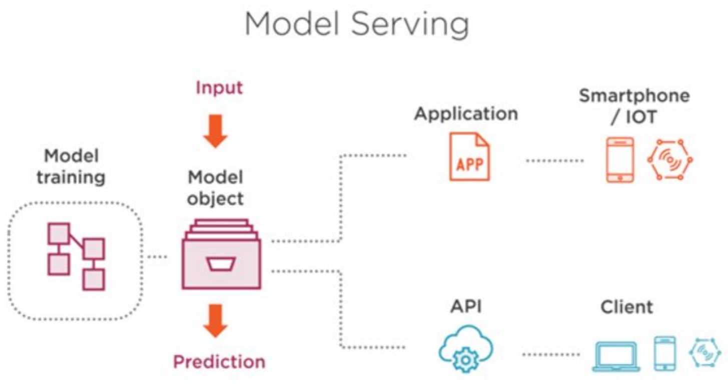
Berkeley
UNIVERSITY OF CALIFORNIA

Background:

Background:

This project utilizes the DistilBertForSequenceClassification sentiment analysis model which is inside of the "glue" dataset.

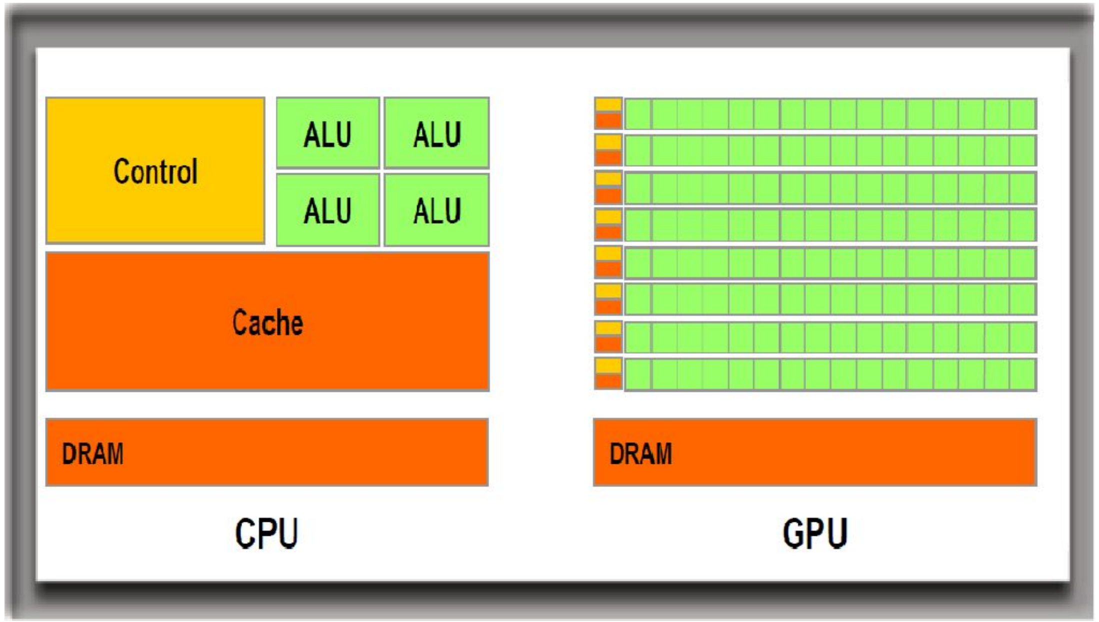
It downloads, or trains the model then serves it within a FASAPI container and runs K6 load testing against a minikube instance.



K6 Load Testing

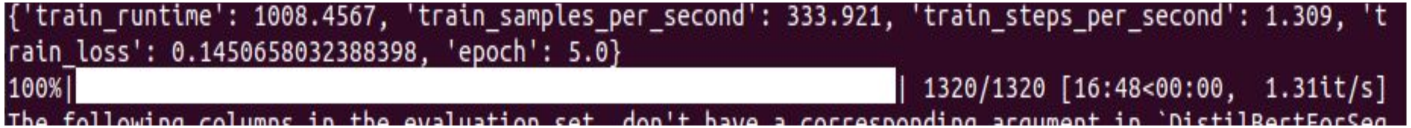
CPU and GPU working together in AI frameworks:

CPUs have larger instruction sets, allowing them to "do more things" than a GPU. However, GPUs generally contain thousands of cores, achieve higher parallelism, and are very good with matrix multiplication, utilized by machine learning, and neural net frameworks. Often times the CPU acts as the "controller" off-loading tasks to the GPU then persisting or evaluating the results.



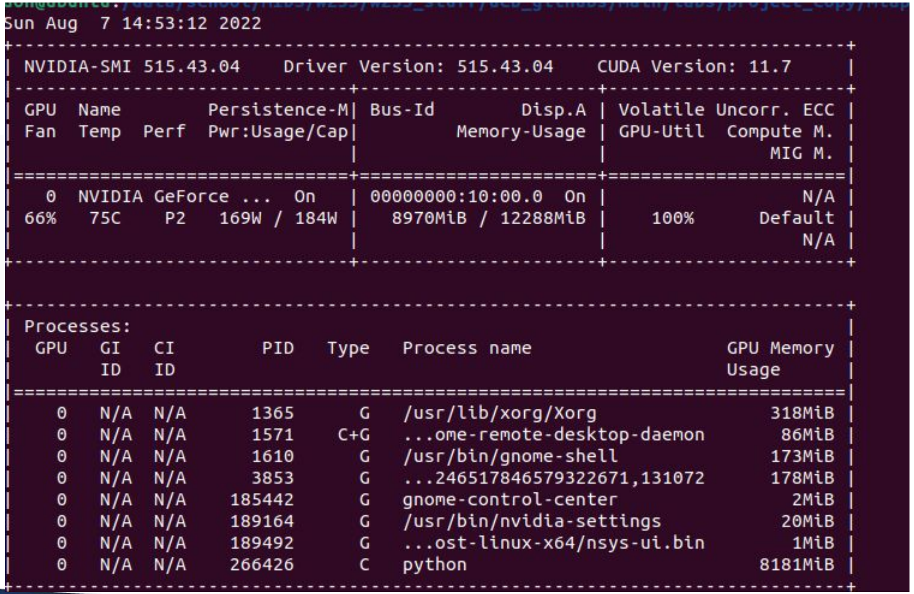
Results: CUDA – GPU train time and CPU/GPU use profile

Train time @ 17 minutes 334 train samples per second.



```
{'train_runtime': 1008.4567, 'train_samples_per_second': 333.921, 'train_steps_per_second': 1.309, 't
rain_loss': 0.1450658032388398, 'epoch': 5.0}
100%|████████████████████████████████████████████| 1320/1320 [16:48<00:00,  1.31it/s]
```

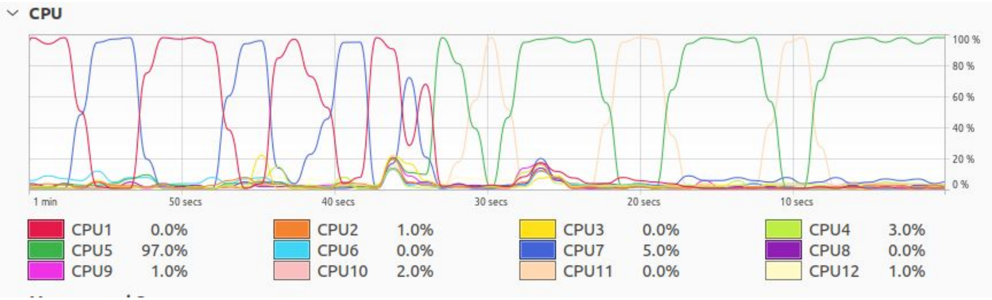
GPU use at 100%, Python Using 8 Gigs of memory



```
Sun Aug  7 14:53:12 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 515.43.04    Driver Version: 515.43.04    CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  On   | 00000000:10:00.0  On |                  N/A |
| 66%   75C    P2   169W / 184W |   8970MiB / 12288MiB |    100%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      1365      G   /usr/lib/xorg/Xorg                318MiB |
|    0   N/A  N/A      1571    C+G   ...ome-remote-desktop-daemon       86MiB |
|    0   N/A  N/A      1610      G   /usr/bin/gnome-shell              173MiB |
|    0   N/A  N/A      3853      G   ...246517846579322671,131072      178MiB |
|    0   N/A  N/A    185442      G   gnome-control-center                2MiB |
|    0   N/A  N/A    189164      G   /usr/bin/nvidia-settings           20MiB |
|    0   N/A  N/A    189492      G   ...ost-linux-x64/nsys-ui.bin        1MiB |
|    0   N/A  N/A    266426      C   python                           8181MiB |
+-----------------------------------------------------------------------------+
```

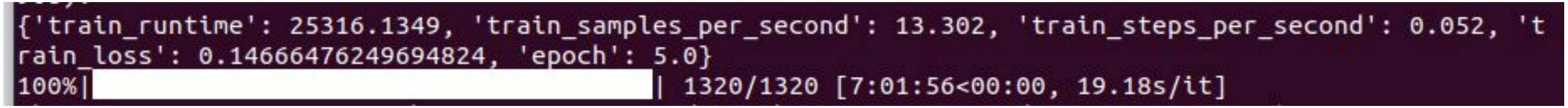
Most CPU cores Unused



🟥	CPU1	0.0%	🟧	CPU2	1.0%	🟨	CPU3	0.0%	🟩	CPU4	3.0%
🟩	CPU5	97.0%	🟦	CPU6	0.0%	🟦	CPU7	5.0%	🟪	CPU8	0.0%
🟪	CPU9	1.0%	🟥	CPU10	2.0%	🟧	CPU11	0.0%	🟨	CPU12	1.0%

Results: CPU train time and CPU/GPU use profile

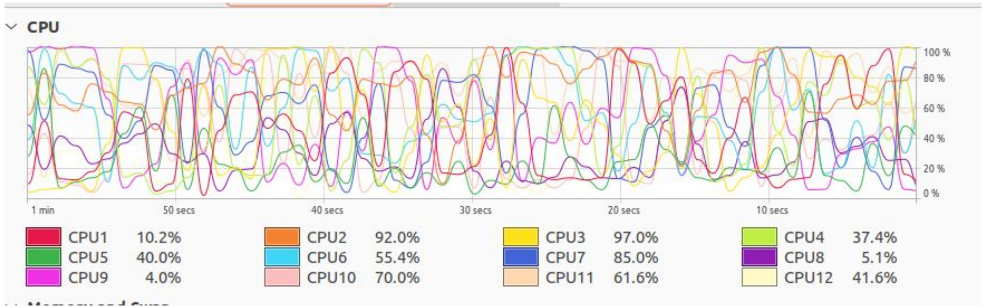
Train time @ 7 hours 13.3 train samples per second



```
{'train_runtime': 25316.1349, 'train_samples_per_second': 13.302, 'train_steps_per_second': 0.052, 't
rain_loss': 0.14666476249694824, 'epoch': 5.0}
100%|                                                              | 1320/1320 [7:01:56<00:00, 19.18s/it]
```

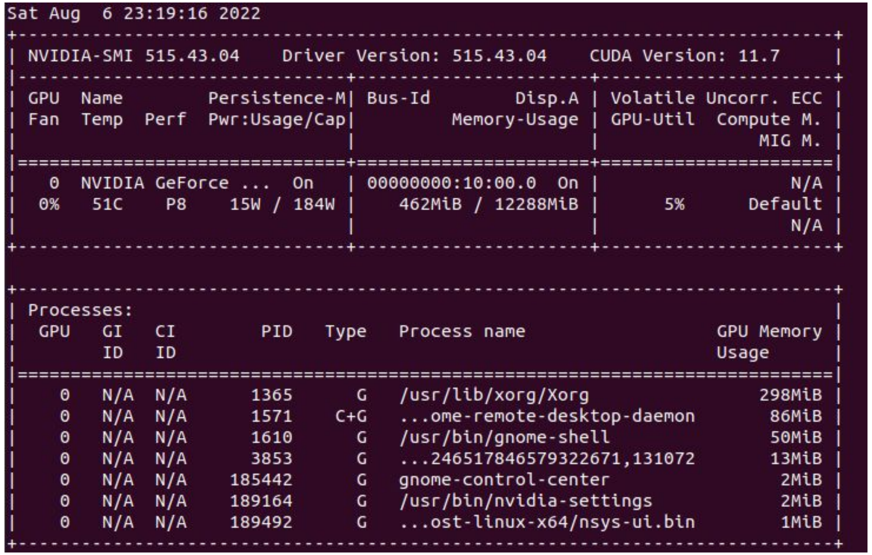
GPU-based training can train the model 24 times in, the amount of time it took the CPU to train the model a single time.

Very High CPU Activity



🟥	CPU1	10.2%	🟧	CPU2	92.0%	🟨	CPU3	97.0%	🟩	CPU4	37.4%
🟩	CPU5	40.0%	🟦	CPU6	55.4%	🟦	CPU7	85.0%	🟪	CPU8	5.1%
🟪	CPU9	4.0%	🟥	CPU10	70.0%	🟧	CPU11	61.6%	🟨	CPU12	41.6%

GPU use at 5%, Python using no memory.

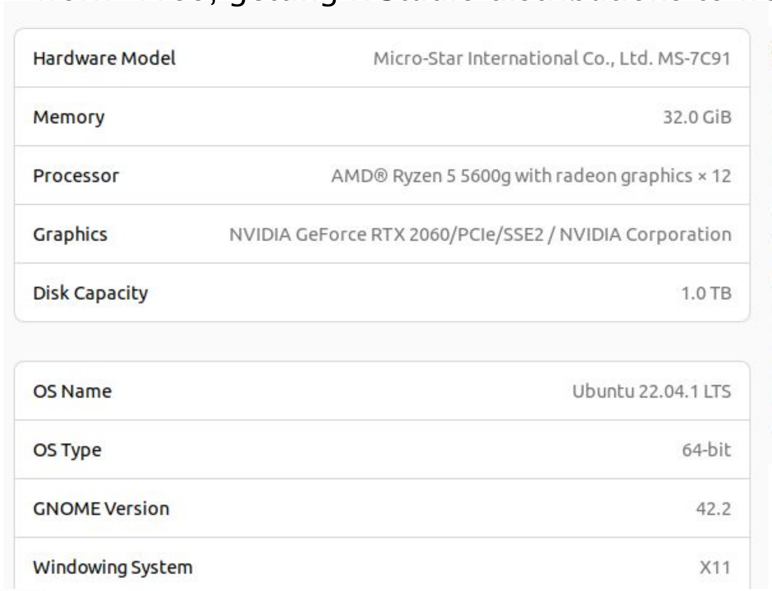


```
Sat Aug  6 23:19:16 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 515.43.04    Driver Version: 515.43.04    CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  On   | 00000000:10:00.0  On |                  N/A |
|  0%   51C    P8    15W / 184W |    462MiB / 12288MiB |      5%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      1365      G   /usr/lib/xorg/Xorg                298MiB |
|    0   N/A  N/A      1571    C+G   ...ome-remote-desktop-daemon       86MiB |
|    0   N/A  N/A      1610      G   /usr/bin/gnome-shell               50MiB |
|    0   N/A  N/A      3853      G   ...246517846579322671,131072       13MiB |
|    0   N/A  N/A    185442      G   gnome-control-center                2MiB |
|    0   N/A  N/A    189164      G   /usr/bin/nvidia-settings            2MiB |
|    0   N/A  N/A    189492      G   ...ost-linux-x64/nsys-ui.bin        1MiB |
+-----------------------------------------------------------------------------+
```

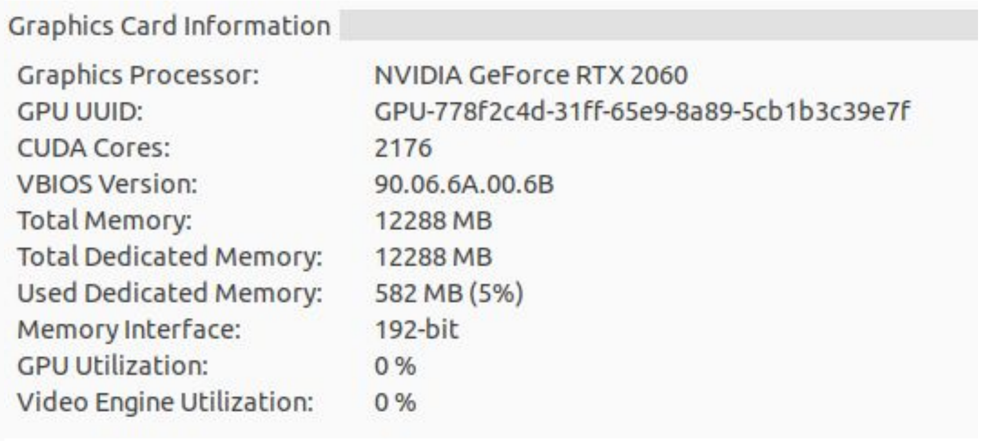
System Specifications:

The system is good, but not ultra high-end. I parted this system together to avoid having to use my MAC M1 given the complications associated with getting packages to run on the M1 architecture. For servers, Linux, the x64 architecture, and the NVIDIA cuda frameworks are widely used. I still use my M1 to terminal into this machine while travelling. It's an uncomplicated way of doing development since many of the packages just work. Also, getting R Studio distributions to work well on an M1 machine can be quite a headache.



Hardware Model	Micro-Star International Co., Ltd. MS-7C91
Memory	32.0 GiB
Processor	AMD® Ryzen 5 5600g with radeon graphics × 12
Graphics	NVIDIA GeForce RTX 2060/PCIe/SSE2 / NVIDIA Corporation
Disk Capacity	1.0 TB

OS Name	Ubuntu 22.04.1 LTS
OS Type	64-bit
GNOME Version	42.2
Windowing System	X11



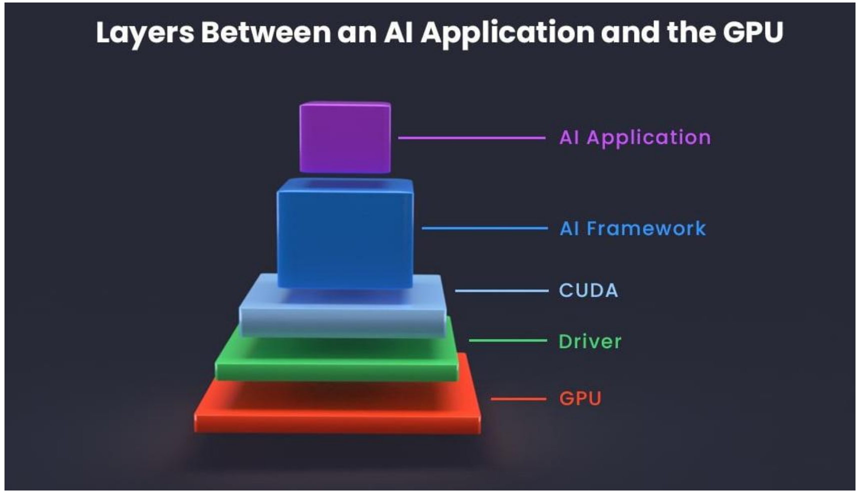
Graphics Card Information

Graphics Processor:	NVIDIA GeForce RTX 2060
GPU UUID:	GPU-778f2c4d-31ff-65e9-8a89-5cb1b3c39e7f
CUDA Cores:	2176
VBIOS Version:	90.06.6A.00.6B
Total Memory:	12288 MB
Total Dedicated Memory:	12288 MB
Used Dedicated Memory:	582 MB (5%)
Memory Interface:	192-bit
GPU Utilization:	0 %
Video Engine Utilization:	0 %

Getting it to work: Be prepared to spend some hours on this.

1. Have an NVIDIA graphics card or device which is compatible with NVIDIA's CUDA package.
2. Install NVIDA drivers.
3. Install the CUDA packages / tools.
4. Test that pytorch can "see" the graphics card and do its work on it:
   ```
   git clone https://github.com/tuneman7/cuda_work  && . ./cuda_work/test_python_cuda.sh
   ```

Some resources on this:
https://tuneman7.github.io/cuda_notes.html
We are also able to access the GPU from within docker images, but that is beyond the scope of this demonstration.



Sample Application:

https://github.com/tuneman7/cuda_work

NOTES:
I could not get cuda / torch to play well under poetry. For training, this project uses python3 virtual environments. For the docker build it still uses poetry.

when the model is trained, the prediction scores will be slightly different after each training so the "assert_almost_equal" statements within the test harness are necessarily removed. You can re-add them after testing it once or twice.

The run.sh script checks for environmental variables, as needed.

ENVIRONMENTAL DEPENDENCIES – in addition to Cuda from previous slide:

Docker, Minikube, K6, Poetry, python3.10-venv

RUNNING:

```
git clone https://github.com/tuneman7/cuda_work
cd cuda_work
. run.sh
```

OR:

```
git clone https://github.com/tuneman7/cuda_work  && cd cuda_work && . run.sh
```

Thank You!