Name: *Yen-Ting Liu*
NetID: *ytliu2*
Section: *AL2*

ECE 408/CS483 Milestone 2 Report

1. Show output of rai running Mini-DNN on the basic GPU convolution implementation for batch size of **1k images**. This can either be a screen capture or a text copy of the running output. Please do not show the build output. (The running output should be everything including and after the line "*Loading fashion-mnist data...Done*").

```
Test batch size: 1000
Loading fashion-mnist data...Done
Loading model...Done
Conv-GPU==
Layer Time: 103.403 ms
Op Time: 13.7382 ms
Conv-GPU==
Layer Time: 228.1 ms
Op Time: 159.083 ms

Test Accuracy: 0.886


real    0m10.014s
user    0m9.783s
sys     0m0.216s
* The build folder has been uploaded to http://s3.amazonaws.com/files.ra
i-project.com/userdata/build-6183fa245876a22e5979b077.tar.gz. The data w
ill be present for only a short duration of time.
```

2. For the basic GPU implementation, list Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 10k images.

Batch Size	Op Time 1	Op Time 2	Total Execution Time	Accuracy
100	*1.37054 ms*	*16.0685 ms*	*1.291 s*	*0.86*
1000	*13.782 ms*	*159.083 ms*	*10.014 s*	*0.886*
10000	*134.958 ms*	*1466.91 ms*	*1 m 38.082 s*	*0.8714*

3. List all the kernels that collectively consumed more than 90% of the kernel time and what percentage of the kernel time each kernel did consume (start with the kernel that consumed the most time, then list the next kernel, until you reach 90% or more).

```
Time(%)      Total Time   Instances         Average         Minimum         Maximum  Name
-------  --------------  ----------  --------------  --------------  --------------  -------------------------
  100.0       170017261           2      85008630.5        12682276       157334985  conv_forward_kernel
```

100.0% conv_forward_kernel

4. List all the CUDA API calls that collectively consumed more than 90% of the API time and what percentage of the API time each call did consume (start with the API call that consumed the most time, then list the next call, until you reach 90% or more).

Time(%)	Total Time	Calls	Average	Minimum	Maximum	Name
39.3	216228287	8	27028535.9	67365	215225976	cudaMalloc
30.9	170047841	8	21255980.1	1299	157339108	cudaDeviceSynchronize
29.6	163186669	10	16318666.9	13609	65926023	cudaMemcpy

39.3% `cudaMalloc`
30.9% `cudaDeviceSynchronize`
29.6% `cudaMemcpy`

5. Explain the difference between kernels and CUDA API calls. Please give an example in your explanation for both.

*(README asks for **API call** versus **kernel launch**, question asks for **kernel**, I'll do all three.)*

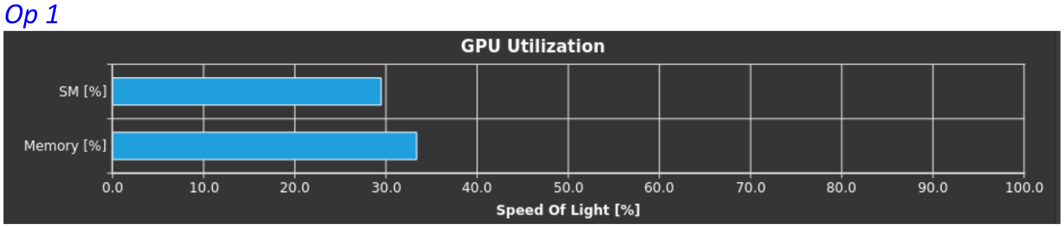
*A **kernel** is a function defined with the `__global__` declaration specifier, which are callable from host, and executed on the device.*

***CUDA runtime API** provides C and C++ functions that execute on the host to manipulate device setup, such as allocate device memory, data transfer, computation on device. The runtime API is built on top of a lower-level C API, the **driver API**, which exposes lower-level concepts: CUDA contexts (similar to processes), and CUDA modules (analogue of dynamically loaded libraries).*

*A **kernel launch** underneath first validates and pushes grid/block/smem/stream parameters, then push function arguments and launch command to buffer, and at last submit command buffer on device to get dispatched to SMs. (CUDA toolkit documentation L.3 Kernel Execution; tests with `nvcc --keep` in `*.cudafe1.c` and `*.cudafe1.stub.c`)*

6. Show a screenshot of the GPU SOL utilization

Op 1



Op 2