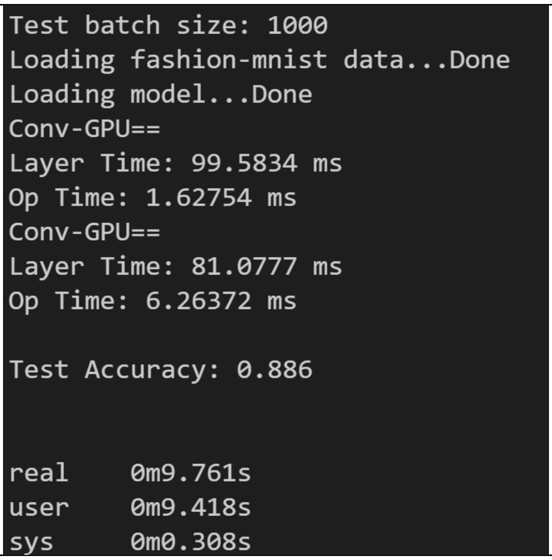
Name: *Sheng-Wei Huang*
NetID: *Swhuang3*
Section: *AL2*

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: 99.5834 ms
Op Time: 1.62754 ms
Conv-GPU==
Layer Time: 81.0777 ms
Op Time: 6.26372 ms

Test Accuracy: 0.886


real    0m9.761s
user    0m9.418s
sys     0m0.308s
```

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	*0.176298ms*	*0.635443ms*	*1.156 s*	*0.86*
1000	*1.62754ms*	*6.26372ms*	*9.761 s*	*0.886*
10000	*16.0021ms*	*62.2321ms*	*95.227 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).

conv_forward_kernel: 100.0% time

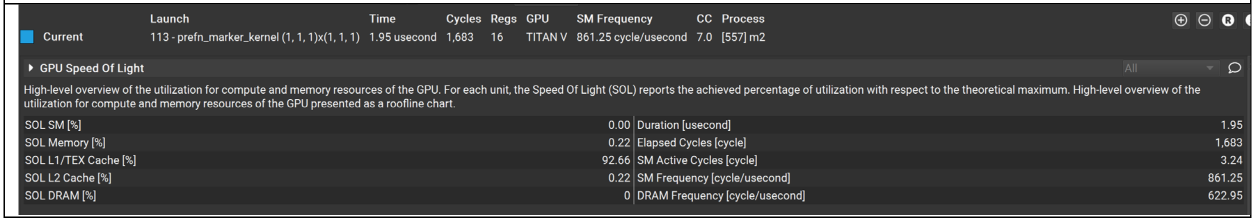
4. List all the CUDA API calls that collectively consumed more than 90% of the API time and what percentage of the API time each call did consume (start with the API call that consumed the most time, then list the next call, until you reach 90% or more).

cudaMemcpy: 84.0%
cudaMalloc: 10.8%

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

Cuda kernels are user defined function with __global__ prefix, which can directly access GPU memory and use GPU to calculate, like __global__ conv_forward_kernel() we defined in this milestone.
Cuda API calls are the api cuda provided, which allow host to allocate or copy data to GPU device, like cudaMemcpy, cudaMalloc.

6. Show a screenshot of the GPU SOL utilization



	Launch	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	113 - prefn_marker_kernel (1, 1, 1)x(1, 1, 1)	1.95 usecond	1,683	16	TITAN V	861.25 cycle/usecond	7.0	[557] m2

GPU Speed Of Light

High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.

SOL SM [%]	0.00	Duration [usecond]	1.95
SOL Memory [%]	0.22	Elapsed Cycles [cycle]	1,683
SOL L1/TEX Cache [%]	92.66	SM Active Cycles [cycle]	3.24
SOL L2 Cache [%]	0.22	SM Frequency [cycle/usecond]	861.25
SOL DRAM [%]	0	DRAM Frequency [cycle/usecond]	622.95