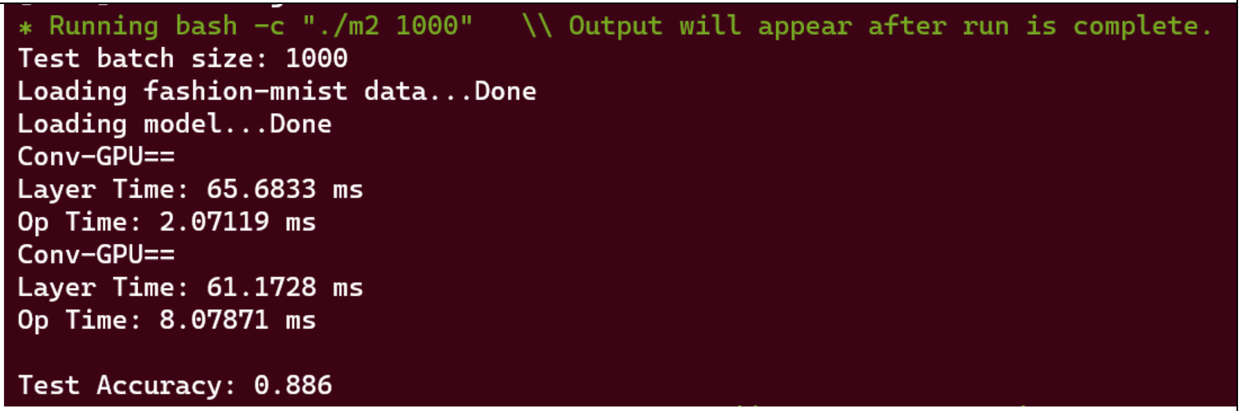
Name: *Marius Juston*
NetID: *Mjuston2*
Section: *AL2*

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*").



```
* Running bash -c "./m2 1000"   \\ Output will appear after run is complete.
Test batch size: 1000
Loading fashion-mnist data...Done
Loading model...Done
Conv-GPU==
Layer Time: 65.6833 ms
Op Time: 2.07119 ms
Conv-GPU==
Layer Time: 61.1728 ms
Op Time: 8.07871 ms

Test Accuracy: 0.886
```

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	*0.219294ms*	*0.818139ms*	*1.037433ms*	*0.86*
1000	*2.07119ms*	*8.07871ms*	*10.1499ms*	*0.886*
10000	*20.4125ms*	*78.6866ms*	*99.0991ms*	*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%

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).

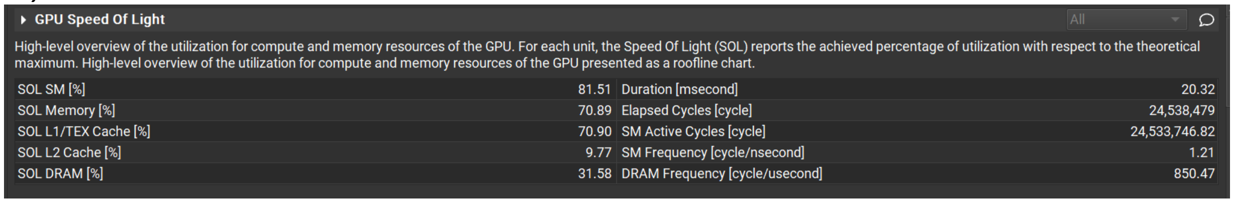
cudaMemcpy 75.1%
cudaMalloc 16.2%

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

The difference between kernels and CUDA API calls is that kernels extend C++ which allows the programmers to interface with threads on the GPU. The CUDA API however, handles more of the device management, such as in initialization, thread management, memory management, etc.… For example, kernels would be the C++ implementation of the vector add but the programmer implements it in such a way that it uses threads efficiently. While the CUDA API handles with cudaMalloc, cudaFree etc… the memory management of the GPU and the background tasks that the programmer does not need to worry about.

6. Show a screenshot of the GPU SOL utilization

Layer 1

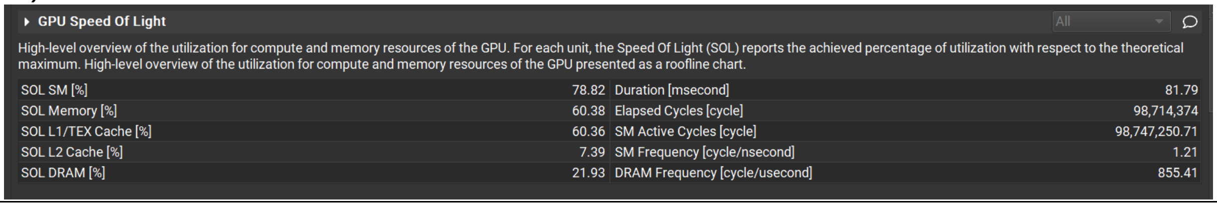


GPU Speed Of Light

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 [%]	81.51	Duration [msecond]	20.32
SOL Memory [%]	70.89	Elapsed Cycles [cycle]	24,538,479
SOL L1/TEX Cache [%]	70.90	SM Active Cycles [cycle]	24,533,746.82
SOL L2 Cache [%]	9.77	SM Frequency [cycle/nsecond]	1.21
SOL DRAM [%]	31.58	DRAM Frequency [cycle/usecond]	850.47

Layer 2



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 [%]	78.82	Duration [msecond]	81.79
SOL Memory [%]	60.38	Elapsed Cycles [cycle]	98,714,374
SOL L1/TEX Cache [%]	60.36	SM Active Cycles [cycle]	98,747,250.71
SOL L2 Cache [%]	7.39	SM Frequency [cycle/nsecond]	1.21
SOL DRAM [%]	21.93	DRAM Frequency [cycle/usecond]	855.41