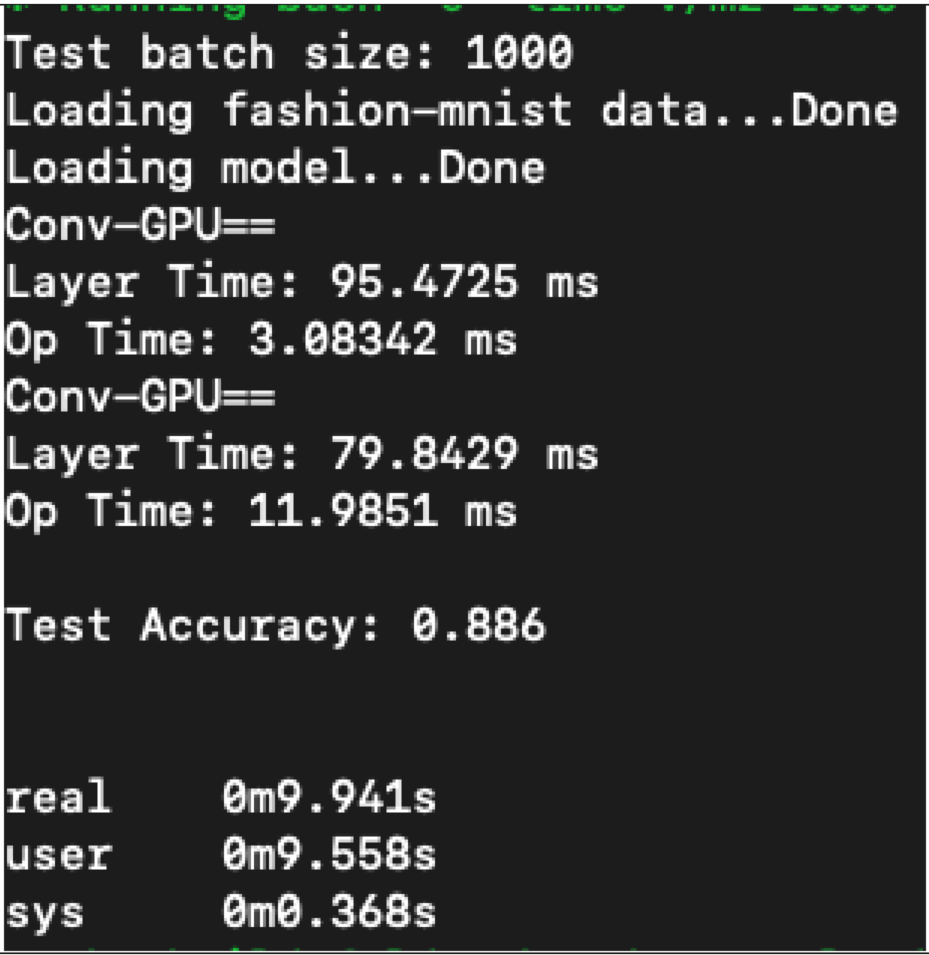
Name: *Aditya Gupta*
NetID: *adityag5*
Section: *AL1*

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: 95.4725 ms
Op Time: 3.08342 ms
Conv-GPU==
Layer Time: 79.8429 ms
Op Time: 11.9851 ms

Test Accuracy: 0.886


real    0m9.941s
user    0m9.558s
sys     0m0.368s
```

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	*0.32 ms*	*1.197 ms*	*1.169 s*	*0.86*
1000	*3.08 ms*	*11.985 ms*	*9.941 s*	*0.886*
10000	*30.44 ms*	*120.418 ms*	*36.117 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 %

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 – 72.4 %

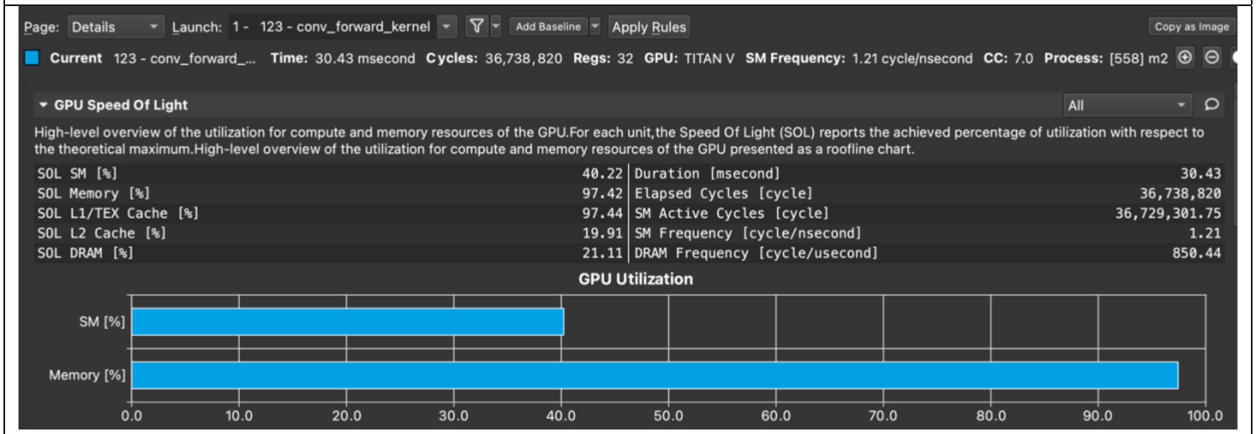
cudaMalloc – 20.2 %

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

CUDA API calls are calls made by the written code into the CUDA driver or runtime libraries whereas the kernel is the function executed on the GPU N times in parallel by N different threads which may contains CUDA API calls.

In this case the conv_forward_kernel is the kernel and functions like cudaMalloc and cudaMemcpy are CUDA API calls.

6. Show a screenshot of the GPU SOL utilization



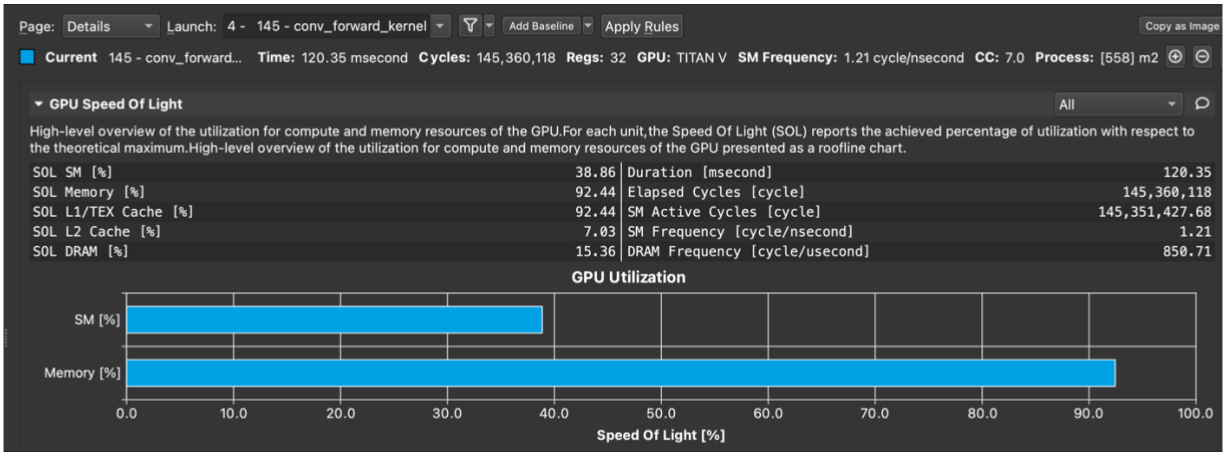
Page: Details Launch: 1 - 123 - conv_forward_kernel Add Baseline Apply Rules Copy as Image

Current 123 - conv_forward_... **Time:** 30.43 msecond **Cycles:** 36,738,820 **Regs:** 32 **GPU:** TITAN V **SM Frequency:** 1.21 cycle/nsecond **CC:** 7.0 **Process:** [558] m2

GPU Speed Of Light All

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 [%]	40.22	Duration [msecond]	30.43
SOL Memory [%]	97.42	Elapsed Cycles [cycle]	36,738,820
SOL L1/TEX Cache [%]	97.44	SM Active Cycles [cycle]	36,729,301.75
SOL L2 Cache [%]	19.91	SM Frequency [cycle/nsecond]	1.21
SOL DRAM [%]	21.11	DRAM Frequency [cycle/usecond]	850.44



Page: Details Launch: 4 - 145 - conv_forward_kernel Add Baseline Apply Rules Copy as Image

Current 145 - conv_forward... **Time:** 120.35 msecond **Cycles:** 145,360,118 **Regs:** 32 **GPU:** TITAN V **SM Frequency:** 1.21 cycle/nsecond **CC:** 7.0 **Process:** [558] m2

GPU Speed Of Light All

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 [%]	38.86	Duration [msecond]	120.35
SOL Memory [%]	92.44	Elapsed Cycles [cycle]	145,360,118
SOL L1/TEX Cache [%]	92.44	SM Active Cycles [cycle]	145,351,427.68
SOL L2 Cache [%]	7.03	SM Frequency [cycle/nsecond]	1.21
SOL DRAM [%]	15.36	DRAM Frequency [cycle/usecond]	850.71