Name: *Wentao Zhang*
NetID: *wentao4*
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: 65.9262 ms
Op Time: 1.62797 ms
Conv-GPU==
Layer Time: 54.2663 ms
Op Time: 6.27431 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	*0.17677 ms*	*0.63500 ms*	*1.163s*	*0.86*
1000	*1.62797 ms*	*6.27431 ms*	*9.691s*	*0.886*
10000	*16.0546 ms*	*62.7971 ms*	*1m34.838s*	*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).

My nsys result shows

```
  100.0       79100647         2     39550323.5       16187218       62913429  conv_forward_kernel
```

*So, the only kernel is **conv_forward_kernel**, it account for 100% of 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).

```
  77.1     1081571065          8     135196383.1          18306       585526607  cudaMemcpy
  15.6      218461729          8      27307716.1          82115       209579309  cudaMalloc
```

*So there are two API: **cudaMemcpy and cudaMalloc** consume more than 90% of running time.*

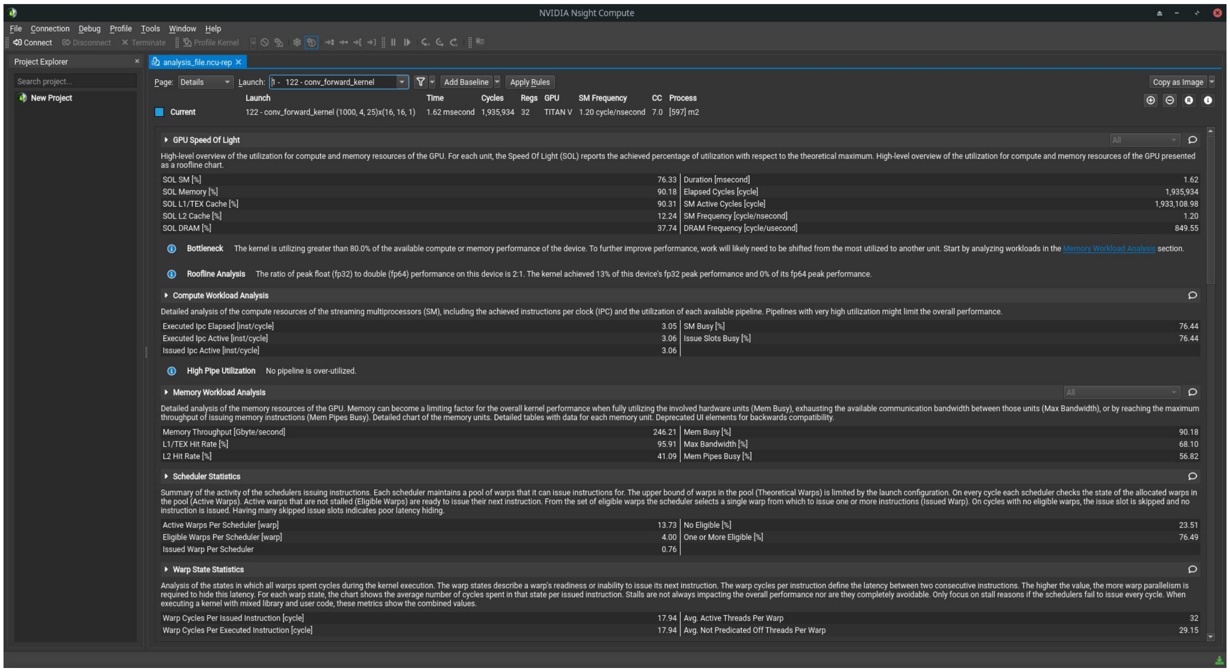
5. Explain the difference between kernels and CUDA API calls. Please give an example in your explanation for both.

*CUDA APIs are used for data transfer, control and resource management. For example, **cudaMemcpy, cudaMemset, cudaMalloc** are CUDA APIs.*

*CUDA kernels are mostly used for efficient computation in parallel , such as **convolution** and **matrix multiplication***

6. Show a screenshot of the GPU SOL utilization

The GPU SOL utilization of two launch:



Page: Details
Launch: 1 - 122 - conv_forward_kernel

	Launch	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	122 - conv_forward_kernel (1000, 4, 25)x(16, 16, 1)	1.62 msecond	1,935,934	32	TITAN V	1.20 cycle/nsecond	7.0	[597] 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 [%]	76.33	Duration [msecond]	1.62
SOL Memory [%]	90.18	Elapsed Cycles [cycle]	1,935,934
SOL L1/TEX Cache [%]	90.31	SM Active Cycles [cycle]	1,933,108.98
SOL L2 Cache [%]	12.24	SM Frequency [cycle/nsecond]	1.20
SOL DRAM [%]	37.74	DRAM Frequency [cycle/usecond]	849.55

Bottleneck The kernel is utilizing greater than 80.0% of the available compute or memory performance of the device. To further improve performance, work will likely need to be shifted from the most utilized to another unit. Start by analyzing workloads in the Memory Workload Analysis section.

Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 2:1. The kernel achieved 13% of this device's fp32 peak performance and 0% of its fp64 peak performance.

Compute Workload Analysis

Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.

Executed Ipc Elapsed [inst/cycle]	3.05	SM Busy [%]	76.44
Executed Ipc Active [inst/cycle]	3.06	Issue Slots Busy [%]	76.44
Issued Ipc Active [inst/cycle]	3.06		

High Pipe Utilization No pipeline is over-utilized.

Memory Workload Analysis

Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit. Deprecated UI elements for backwards compatibility.

Memory Throughput [Gbyte/second]	246.21	Mem Busy [%]	90.18
L1/TEX Hit Rate [%]	95.91	Max Bandwidth [%]	68.10
L2 Hit Rate [%]	41.09	Mem Pipes Busy [%]	56.82

Scheduler Statistics

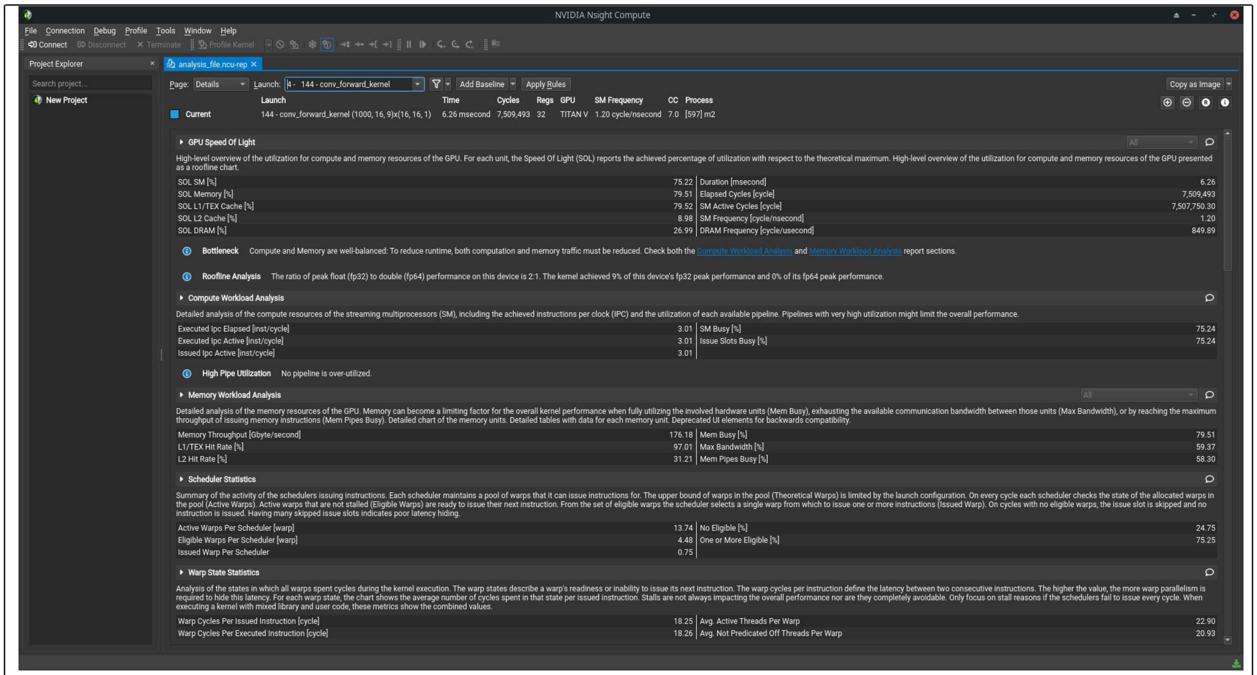
Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.

Active Warps Per Scheduler [warp]	13.73	No Eligible [%]	23.51
Eligible Warps Per Scheduler [warp]	4.00	One or More Eligible [%]	76.49
Issued Warp Per Scheduler	0.76		

Warp State Statistics

Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle. When executing a kernel with mixed library and user code, these metrics show the combined values.

Warp Cycles Per Issued Instruction [cycle]	17.94	Avg. Active Threads Per Warp	32
Warp Cycles Per Executed Instruction [cycle]	17.94	Avg. Not Predicated Off Threads Per Warp	29.15



	Launch	Time	Cycles	Regs	GPU	SM Frequency	CC	Process
Current	144 - conv_forward_kernel (1000, 16, 9)x(16, 16, 1)	6.26 msecond	7,509,493	32	TITAN V	1.20 cycle/nsecond	7.0	[597] 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 [%]	75.22	Duration [msecond]	6.26
SOL Memory [%]	79.51	Elapsed Cycles [cycle]	7,509,493
SOL L1/TEX Cache [%]	79.52	SM Active Cycles [cycle]	7,507,750.30
SOL L2 Cache [%]	8.98	SM Frequency [cycle/nsecond]	1.20
SOL DRAM [%]	26.99	DRAM Frequency [cycle/usecond]	849.89

Bottleneck Compute and Memory are well-balanced: To reduce runtime, both computation and memory traffic must be reduced. Check both the Compute Workload Analysis and Memory Workload Analysis report sections.

Roofline Analysis The ratio of peak float (fp32) to double (fp64) performance on this device is 2:1. The kernel achieved 9% of this device's fp32 peak performance and 0% of its fp64 peak performance.

Compute Workload Analysis

Detailed analysis of the compute resources of the streaming multiprocessors (SM), including the achieved instructions per clock (IPC) and the utilization of each available pipeline. Pipelines with very high utilization might limit the overall performance.

Executed Ipc Elapsed [inst/cycle]	3.01	SM Busy [%]	75.24
Executed Ipc Active [inst/cycle]	3.01	Issue Slots Busy [%]	75.24
Issued Ipc Active [inst/cycle]	3.01		

High Pipe Utilization No pipeline is over-utilized.

Memory Workload Analysis

Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy). Detailed chart of the memory units. Detailed tables with data for each memory unit. Deprecated UI elements for backwards compatibility.

Memory Throughput [Gbyte/second]	176.18	Mem Busy [%]	79.51
L1/TEX Hit Rate [%]	97.01	Max Bandwidth [%]	59.37
L2 Hit Rate [%]	31.21	Mem Pipes Busy [%]	58.30

Scheduler Statistics

Summary of the activity of the schedulers issuing instructions. Each scheduler maintains a pool of warps that it can issue instructions for. The upper bound of warps in the pool (Theoretical Warps) is limited by the launch configuration. On every cycle each scheduler checks the state of the allocated warps in the pool (Active Warps). Active warps that are not stalled (Eligible Warps) are ready to issue their next instruction. From the set of eligible warps the scheduler selects a single warp from which to issue one or more instructions (Issued Warp). On cycles with no eligible warps, the issue slot is skipped and no instruction is issued. Having many skipped issue slots indicates poor latency hiding.

Active Warps Per Scheduler [warp]	13.74	No Eligible [%]	24.75
Eligible Warps Per Scheduler [warp]	4.48	One or More Eligible [%]	75.25
Issued Warp Per Scheduler	0.75		

Warp State Statistics

Analysis of the states in which all warps spent cycles during the kernel execution. The warp states describe a warp's readiness or inability to issue its next instruction. The warp cycles per instruction define the latency between two consecutive instructions. The higher the value, the more warp parallelism is required to hide this latency. For each warp state, the chart shows the average number of cycles spent in that state per issued instruction. Stalls are not always impacting the overall performance nor are they completely avoidable. Only focus on stall reasons if the schedulers fail to issue every cycle. When executing a kernel with mixed library and user code, these metrics show the combined values.

Warp Cycles Per Issued Instruction [cycle]	18.25	Avg. Active Threads Per Warp	22.90
Warp Cycles Per Executed Instruction [cycle]	18.26	Avg. Not Predicated Off Threads Per Warp	20.93