Name: *Mohit Mahajan*
NetID: *mohitm3*
Section: *CS483 AL2 (67070)*

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*").

```
[100%] Built target final
* Running bash -c "time ./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: 63.8349 ms
Op Time: 1.66012 ms
Conv-GPU==
Layer Time: 52.7733 ms
Op Time: 6.37031 ms

Test Accuracy: 0.886


real    0m10.206s
user    0m9.924s
sys     0m0.284s
* The build folder has been uploaded to http://s3.amazonaws.com/files.rai-project.
```

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	*0.987 ms*	*0.643 ms*	*3.214 sec*	*0.86*
1000	*1.66 ms*	*6.37 ms*	*10.206 sec*	*0.886*
10000	*16.24 ms*	*6583.21 ms*	*1m 45.8 sec*	*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).

As per the CUDA kernel statistics reported in the M2-nsyscheck.output file included in the submission, Following are the kernels that consume 90% of the kernel time:

- `conv_forward_kernel` *(~ 100% of the 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).

As per the CUDA API statistics reported in the M2-nsyscheck.output file included in the submission, Following are the API calls that consume 90% of the API time:

- `cudaMemcpy` *(65%)*
- `cudaMalloc` *(22%)*
- `cudaDeviceSynchronize` *(11%)*

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

Kernel is the method that runs the computation for the intended parallel operation. This is the core functionality of any GPU parallel operation, that manipulates inputs and generates the desired output. Kernel is invoked at the thread level.

Example: `conv_forward_kernel` *developed in the milestone-2 computes the output feature map for a tile of input image using all input feature maps.*

CUDA API calls enable the program to launch, control and coordinate the kernel methods in a cuda program. Running a kernel requires some amount of pre & post processing like (de-)allocation of memory and processing resources, copying the necessary data between GPU and CPU memory, catch-errors etc. Thus, a typical GPU parallel program comprises a logical sequence of API calls and kernels.

Example: `cudaMalloc`, `cudaMemcpy` *used in the milestone-2 allocate GPU memory and copy the data from CPU to GPU before launching the* `conv_forward_kernel`, *these are CUDA API calls.*

6. Show a screenshot of the GPU SOL utilization