ECE 408 FINAL PROJECT REPORT (MILESTONE 3)

Ahmad Syafiq bin Shamsul Bahari (shamsul2)
Anirban Banerjee (banerj10)
Srujun Thanmay Gupta (sgupta80)

1.1: Run the Baseline Forward Pass

We got result:

```
EvalMetric: {'accuracy': 0.8673}
9.98user 2.83system 0:04.12elapsed 310%CPU
```

1.2: Run the baseline GPU implementation

We got result:

```
EvalMetric: {'accuracy': 0.8673}
1.73user 0.92system 0:02.19elapsed 121%CPU
```

The CPU load is considerably less in this run, indicating that most work happened on the GPU instead.

1.3 Generate a NVPROF Profile

Most time consuming kernel calls:

Time (%)	**Time (ms)**	**Kernel Name**
37.00%	49.981ms	`void cudnn::detail::implicit_convolve_sgemm(...)`
28.65%	38.704ms	`sgemm_sm35_ldg_tn_128x8x256x16x32`
14.35%	19.385ms	`void cudnn::detail::activation_fw_4d_kernel(...)`
10.70%	14.451ms	`void cudnn::detail::pooling_fw_4d_kernel(...)`
		...

From this list, it seems that the GPU spends the bulk of the time in 2 kernels, the actual convolution kernel and a large single precision floating point general matrix multiplication (SGEMM).

2.1 Simple CPU forward implementation

Using the baseline CPU implementation, we got the following output.

```
✱ Running python /src/m2.1.py
New Inference
Loading fashion-mnist data... done
Loading model... done
Op Time: 11.722999
Correctness: 0.8562 Model: ece408-high
```

This matches the given Op Time and Correctness from the README.

3.1 Simple GPU forward implementation

Using the baseline GPU implementation, we got the following output.

```
✱ Running time python /src/m3.1.py
batches = 10000
fmaps = 50
channels = 1
img_height = 28
img_width = 28
kernel_size = 5

grid(10000, 50, 1)
block(24, 24, 1)
Op Time: 0.400895
Correctness: 0.8562 Model: ece408-high
2.12user 1.02system 0:02.64elapsed 119%CPU (0avgtext+0avgdata
908044maxresident)k
```

NVPROF Profile:

Time (%)	**Time (ms)**	**Kernel Name**
80.51%	428.66ms	`mxnet::op::forward_kernel(...)`
7.38%	39.32ms	`sgemm_sm35_ldg_tn_128x8x256x16x32`
3.67%	19.56ms	`void mshadow::cuda::MapPlanLargeKernel<...>`
3.64%	19.38ms	`void cudnn::detail::activation_fw_4d_kernel<...>`
		...

The kernel code that we have implemented is still quite time consuming since we have not implemented any optimizations like shared memory usage or GEMM matrix multiplication.

Teamwork division:

All three of us sat together while working on this milestone. We discussed the structure of the tensors and how we could create the CUDA Grid and Blocks for the forward propagation of the CNN. We then sat together and programmed the kernel functionality.