M1.1 Milestone 1.1

```
* Running python /src/m1.1.py
...
EvalMetric: {'accuracy': 0.8673}
0:06.77 elapsed
```

M1.2/1.3 Milestone 1.2/1.3

The GPU spends most of its compute time in
* **implicit_convolve_sgemm**
* **sgemm_sm35_ldg_tn_128x8x256x16x32**, and
* **activation_fw_4d_kernel**.

Most of the compute time is spend performing convolution. There are also memory API calls that take more time than computation.

```
* Running nvprof python /src/m1.2.py
...
==305== Profiling result:
Time(%)      Time  Calls       Avg       Min       Max  Name
36.73%   50.515ms      1  50.515ms  50.515ms  50.515ms  implicit_convolve_sgemm
28.66%   39.409ms      1  39.409ms  39.409ms  39.409ms  sgemm_sm35_ldg_tn_128x8x256
14.10%   19.396ms      2  9.6979ms  461.70us  18.934ms  activation_fw_4d_kernel
...
==305== API calls:
Time(%)      Time  Calls       Avg       Min       Max  Name
 46.82%  1.85071s     18  102.82ms  17.325us  925.01ms  cudaStreamCreateWithFlags
 28.48%  1.12580s     10  112.58ms     949ns  320.49ms  cudaFree
 20.78%  821.31ms     24  34.221ms  236.10us  814.20ms  cudaMemGetInfo
...
```

M2.1 Milestone 2.1

```
* Running python /src/m2.1.py
Loading fashion-mnist data... done
Loading model... done
Op Time: 12.146829
Correctness: 0.8562 Model: ece408-high

* Running python m2.1.py ece408-low 10000
Loading fashion-mnist data... done
Loading model... done
Op Time: 12.801514
Correctness: 0.629 Model: ece408-low
```

Team Member Contributions

Sujay: Milestone 2 code
Tanishq: Milestone 2 pdf
Gordon: pdf edits

M3.1 Milestone 3.1

```
* Running /usr/bin/time python m3.1.py ece408-high 10000
New Inference
Loading fashion-mnist data... done
Loading model... done
Op Time: 0.509298
```

```
Correctness: 0.8562 Model: ece408-high
2.15user 1.05system 0:02.72elapsed 117%CPU (0avgtext+0avgdata 907624maxresident)k
0inputs+3136outputs (0major+157453minor)pagefaults 0swaps



* Running nvprof python m3.1.py ece408-high 10000
New Inference
Loading fashion-mnist data... done
==309== NVPROF is profiling process 309, command: python m3.1.py ece408-high 10000
Loading model... done
Op Time: 0.556559
Correctness: 0.8562 Model: ece408-high
==309== Profiling application: python m3.1.py ece408-high 10000
==309== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
83.62%  536.96ms         1  536.96ms  536.96ms  536.96ms  void mxnet::op::
    forward_kernel<mshadow::gpu, float>(float*, mxnet::op::
    forward_kernel<mshadow::gpu, float> const *, mxnet::op::
    forward_kernel<mshadow::gpu, float> const , int, int, int, int, int, int)
 6.11%  39.268ms         1  39.268ms  39.268ms  39.268ms  sgemm_sm35_
    ldg_tn_128x8x256x16x32
 3.05%  19.565ms         1  19.565ms  19.565ms  19.565ms  void mshadow::cuda::
    MapPlanLargeKernel<mshadow::sv::saveto, int=8, int=1024, mshadow::expr::
    Plan<mshadow::Tensor<mshadow::gpu, int=4, float>, float>, mshadow::expr::
    Plan<mshadow::expr::BinaryMapExp<mshadow::op::mul, mshadow::expr::
    ScalarExp<float>, mshadow::Tensor<mshadow::gpu, int=4, float>, float, int=1>,
    float>>(mshadow::gpu, unsigned int, mshadow::Shape<int=2>, int=4, int)
 3.02%  19.393ms         2  9.6963ms  461.59us  18.931ms  void cudnn::detail::
    activation_fw_4d_kernel<float, float, int=128, int=1, int=4, cudnn::detail::
    tanh_func<float>>(cudnnTensorStruct, float const *, cudnn::detail::activation_
    fw_4d_kernel<float, float, int=128, int=1, int=4,
    cudnn::detail::tanh_func<float>>, cudnnTensorStruct*, float,
    cudnnTensorStruct*, int, cudnnTensorStruct*)
 <snip>
==309== API calls:
Time(%)      Time   Calls       Avg       Min       Max  Name
 41.57%  1.86458s      18  103.59ms  17.408us  932.13ms  cudaStreamCreateWithFlags
 25.33%  1.13601s      10  113.60ms  1.0270us  325.13ms  cudaFree
 18.41%  825.64ms      23  35.898ms  236.27us  818.92ms  cudaMemGetInfo
 12.41%  556.53ms       1  556.53ms  556.53ms  556.53ms  cudaDeviceSynchronize
 <snip>
```

Team Member Contributions
Sujay: Milestone 3 pdf
Tanishq: pdf edits
Gordon: Milestone 3 code

Optimization Approach and Results

As stated in section 3.1, our baseline GPU runtime was 509.298 ms.

We did the following optimizations:

- Shared memory tiles
  By moving sections of X into per-block shared memory, we reduce the amount of global memory accesses done. Global memory accesses are slow; by doing faster shared memory accesses instead, our runtime was reduced from 509.298 ms to 94.931 ms, an 81% decrease in runtime.

- Mask in constant memory
  The convolution mask is relatively small and never changes, so it's a good candidate to move to constant memory. Constant memory is also much faster to access than global memory. By making constant memory accesses instead of global memory accesses when reading the convolution mask, we reduced our runtime from 94.931 ms (GPU implementation with shared memory tiles) to 13.710 ms, an 86% decrease in runtime.

- Unroll the outermost loop in the kernel
  Loop unrolling reduces the number of costly branch instructions in code and can be easily done with an nvcc pragma. We experimented and found that an unroll factor of 15 gave the greatest performance increase, although the performance gain was small compared to memory optimizations. We reduced our runtime from 13.710 ms (GPU implementation with memory optimizations) to 13.234 ms, a 3.5% decrease in runtime.

- Hardcode values
  We thought that by hardcoding input and output dimensions, we could reduce the number of instructions generated. Hardcoding values would eliminate loads from registers and place the value directly in the instruction itself. We could also pre-compute some array indices at compile time instead of at runtime. We noticed a tiny performance gain here; runtime decreased from 13.234 ms to 13.225 ms (<0.1% decrease).

References

https://devtalk.nvidia.com/default/topic/384389/loop-unrolling/

Team Member Contributions

Sujay/Tanishq: Final optimized code
Gordon: Final report