ECE408 - Applied Parallel Programming
Final Report

Neil Singh (ngsingh2)
Patrick McMahon (pfmcmah2)
Matthew Krikorian (krikorn2)

November 29, 2017

Milestone 1

Milestone 1 of this project was mostly setting up our test environment and preparing it to start developing the later parts of this project.

Running Time of m1.1.py

For m1.1.py, we had an accuracy of 0.8673, and an elapsed running time of 9.76 seconds.

Running Time of m1.2.py

For m1.2.py, we had an accuracy of 0.8673, and an elapsed running time of 3.19 seconds.

Most Time Consuming Kernels

Some of the most time consuming kernels when running the nvprof profile were the*implicit_convolve_sgemm* kernel, taking up 36.96% of the execution time, the *activation_fw_4d_kernel* kernel, taking up 14.33% of the execution time, and the *pooling_fw_4d_kernel* kernel, which took up 10.69% of the programs execution time.

Milestone 2

In this milestone of the project, we were tasked with writing sequential convolution code. Using the skeleton from our textbook, and the corresponding data accesses, we successfully implemented the function below.

```
{
    const int B = x.shape_[0];
    const int M = y.shape_[1];
    const int C = x.shape_[1];
    const int H = x.shape_[2];
    const int W = x.shape_[3];
    const int K = k.shape_[3];

    int H_out = H - K + 1;
    int W_out = W - K + 1;

    for (int b = 0; b < B; ++b)
    {
      for (int m = 0; m < M; m++)
      {
        for (int h = 0; h < H_out; h++)
        {
          for (int w = 0; w < W_out; w++)
          {
            y[b][m][h][w] = 0;

            for (int c = 0; c < C; c++)
            {
              for (int p = 0; p < K; p++)
              {
                for (int q = 0; q < K; q++)
                {
                  y[b][m][h][w] += x[b][c][h + p][w + q] * k[m][c][p][q];
                }
              }
            }
          }
        }
      }
    }
}
```

Figure 1: Our CPU implementation of the convolutional layer in MxNet

Our Results For Respective Models

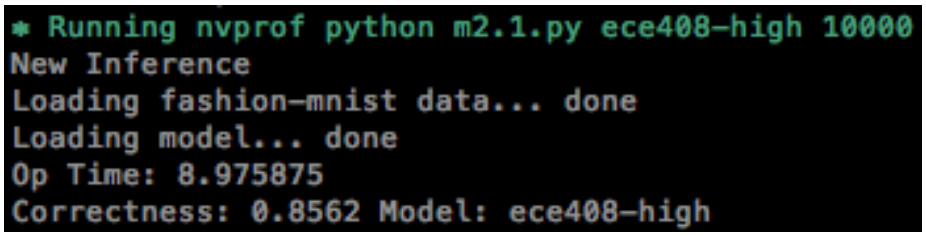

Figure 2: Accuracy and elapsed time for high correctness and sample size 10000

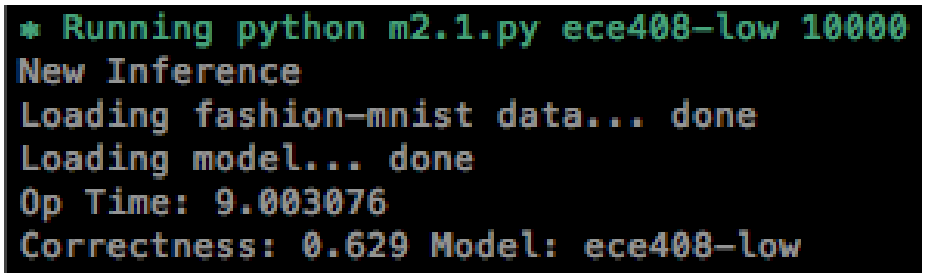

Figure 3: Accuracy and elapsed time for low correctness and sample size 10000

Milestone 3

In this milestone we were tasked with writing a baseline paralllel convolution kernel that had to pass several baseline tests. This implementation is very sub-optimal, as it uses a TILE_WIDTH of 1.

```
{
    const int H_out = H - K + 1;
    const int W_out = W - K + 1;

    #define y4d(i3,i2,i1,i0) y[(i3) * (M * H_out * W_out) + (i2)*(H_out
        * W_out) + (i1)*(W_out) + i0]
    #define x4d(i3,i2,i1,i0) x[(i3) * (C * H * W) + (i2)*(H * W) +
        (i1)*(W) + i0]
    #define k4d(i3,i2,i1,i0) k[(i3) * (C * K * K) + (i2)*(K * K) +
        (i1)*(K) + i0]

    int W_grid = ceil(W_out / (float)TILE_WIDTH);
    int H_grid = ceil(H_out / (float)TILE_WIDTH);
    int n, m, h, w, c, p, q;
    n = blockIdx.x;
    m = blockIdx.y;
    h = blockIdx.z / W_grid + threadIdx.y;
    w = blockIdx.z % W_grid + threadIdx.x;

    float acc = 0;
    for (c = 0; c < C; c++) {
        for (p = 0; p < K; p++) {
            for (q = 0; q < K; q++) {
                if (h+p < H && w+q < W)
                    acc += x4d(n, c, h+p, w+q) * k4d(m, c, p, q);
            }
        }
    }
    y4d(n, m, h, w) = acc;

    #undef y4d
    #undef x4d
    #undef k4d
}
```

Figure 4: Our GPU implementation of the convolutional layer in MxNet

Nvprof GPU Profile

```
* Running nvprof python m3.1.py
Loading fashion-mnist data... done
==311== NVPROF is profiling process 311, command: python m3.1.py
Loading model... done
Op Time: 21.654475
Correctness: 0.8562 Model: ece408-high
==311== Profiling application: python m3.1.py
==311== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
99.60%  21.6544s         1  21.6544s  21.6544s  21.6544s  void mxnet::op::forward_kernel
```

Figure 5: Nvprof GPU profile for our forward kernel

Team Contributions

We all met up together and finished the sequential code for the convolutional layer in person. We also met up and wrote the parallel code together after meeting up in person, as well as crafting this report together for submission.