Team: WE WANT 508
Members: *Haoran Qi* (haoranq2), *Yao Xu* (yaoxu5), and *Wenhan Zhao* (wenhanz3)

ECE 408 Course Project Report (2nd Draft)

Milestone 1

1.1 Baseline Forward Pass

In this stage, CPU code for baseline forward pass was run to verify successful establishment of rai and confirm expected output.

RAI binary version 0.2.18 for Linux was downloaded from the URL in README.md. File ~/.rai_profile was modified with authentication tokens received from email. Folder 2017fa_ece408_project was git cloned by inputting the following command in terminal:

git clone https://github.com/webgpu/2017fa_ece408_project.git

rai_build.yml was modified to measure the elapsed time. The command executing m1.1.py was changed into "/usr/bin/time python /src/m.1.1.py".

After inputting the command "./rai -p 2017fa_ece408_project", result was showed as below.

```
* Running /usr/bin/time python m1.1.py
New Inference
Loading fashion-mnist data... done
Loading model... done
EvalMetric: {'accuracy': 0.8673}
10.04user 2.99system 0:04.17elapsed 312%CPU (0avgtext+0avgdata
1634072maxresident)k
0inputs+2624outputs (0major+29730minor)pagefaults 0swaps
*     The     build     folder     has     been     uploaded     to
http://s3.amazonaws.com/files.rai-project.com/userdata/build-
877e8f2f-79c5-43be-b26b-a0ffef87bf47.tar.gz. The data will be
present for only a short duration of time.
```

The accuracy of 0.8673 was confirmed as mentioned in README.md. The elapsed time was 4.17 seconds.

1.2 Baseline GPU Implementation

In this stage, rai_build.yml was modified to run the baseline code by GPU. As guided by README.md, image inputted was changed, count of GPU was set to 1 and python file to be execute was changed to m1.2.py.

After running the project, result was shown as below:

```
* Running /usr/bin/time python m1.2.py
New Inference
Loading fashion-mnist data... done
Loading model...[01:56:20] src/operator/./././cudnn_algoreg-inl.h:112: Running performance tests to find the best convolution algorithm, this can take a while... (setting env variable MXNET_CUDNN_AUTOTUNE_DEFAULT to 0 to disable)
done
EvalMetric: {'accuracy': 0.8673}
1.75user 1.05system 0:02.29elapsed 122%CPU (0avgtext+0avgdata 916960maxresident)k
0inputs+3136outputs (0major+159473minor)pagefaults 0swaps
* The build folder has been uploaded to http://s3.amazonaws.com/files.rai-project.com/userdata/build-000799f5-9672-4385-b54a-f67eb66e1988.tar.gz. The data will be present for only a short duration of time.
```

The accuracy was still 0.8673 while the elapsed time was now 2.29 seconds.

1.3 Generate a NVPROF Profile

In this stage, a profile was generated by changing the code executing python files into “nvprof python m1.2.py”. The generated profile shows statistics about CUDA kernels and API calls, including the percentage of time consumed, total time, average, maximum and minimum time for every kernels or API functions.

```
* Running nvprof python m1.2.py
New Inference
Loading fashion-mnist data... done
==311== NVPROF is profiling process 311, command: python m1.2.py
Loading model...[02:08:15] src/operator/./././cudnn_algoreg-inl.h:112: Running performance tests to find the best convolution algorithm, this can take a while... (setting env variable MXNET_CUDNN_AUTOTUNE_DEFAULT to 0 to disable)
done
EvalMetric: {'accuracy': 0.8673}
==311== Profiling application: python m1.2.py
==311== Profiling result:
Time(%)     Time    Calls      Avg      Min      Max  Name
37.04%   50.447ms                1   50.447ms   50.447ms   50.447ms   void cudnn::detail::implicit_convolve_sgemm<float, int=1024, int=5, int=5, int=3, int=3, int=3, int=1, bool=1, bool=0, bool=1>(int, int, int, float const *, int, cudnn::detail::implicit_convolve_sgemm<float, int=1024, int=5, int=5, int=3, int=3, int=3, int=1, bool=1, bool=0, bool=1>*, float const *, kernel_conv_params, int, float, float, int, float const *, float const *, int, int)
```

```
28.80%    39.223ms                  1    39.223ms    39.223ms    39.223ms
sgemm_sm35_ldg_tn_128x8x256x16x32
14.23%    19.381ms              2    9.6904ms    460.73us    18.920ms    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*)
 10.64%    14.493ms              1    14.493ms    14.493ms    14.493ms    void
cudnn::detail::pooling_fw_4d_kernel<float,                          float,
cudnn::detail::maxpooling_func<float,           cudnnNanPropagation_t=0>,
int=0>(cudnnTensorStruct,            float            const            *,
cudnn::detail::pooling_fw_4d_kernel<float,                          float,
cudnn::detail::maxpooling_func<float,  cudnnNanPropagation_t=0>,  int=0>,
cudnnTensorStruct*, cudnnPoolingStruct, float, cudnnPoolingStruct, int,
cudnn::reduced_divisor, float)
  4.55%  6.2003ms       13  476.94us  1.5360us  4.2538ms  [CUDA memcpy HtoD]
  2.73%    3.7179ms                  1    3.7179ms    3.7179ms    3.7179ms
sgemm_sm35_ldg_tn_64x16x128x8x32
0.82%    1.1201ms              1    1.1201ms    1.1201ms    1.1201ms    void
mshadow::cuda::SoftmaxKernel<int=8,                                  float,
mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu,  int=2,  float>,  float>,
mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu,     int=2,     float>,
float>>(mshadow::gpu, int=2, unsigned int)
  0.55%    754.54us             12    62.878us    2.0800us    380.79us    void
mshadow::cuda::MapPlanKernel<mshadow::sv::saveto,                   int=8,
mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu,  int=2,  float>,  float>,
mshadow::expr::Plan<mshadow::expr::ScalarExp<float>,
float>>(mshadow::gpu, unsigned int, mshadow::Shape<int=2>, int=2)
0.32%    437.01us              2    218.51us    16.959us    420.06us    void
mshadow::cuda::MapPlanKernel<mshadow::sv::plusto,                   int=8,
mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu,  int=2,  float>,  float>,
mshadow::expr::Plan<mshadow::expr::Broadcast1DExp<mshadow::Tensor<mshadow
::gpu, int=1, float>, float, int=2, int=1>, float>>(mshadow::gpu, unsigned
int, mshadow::Shape<int=2>, int=2)
  0.29%    394.46us                  1    394.46us    394.46us    394.46us
sgemm_sm35_ldg_tn_32x16x64x8x16
0.02%    23.488us              1    23.488us    23.488us    23.488us    void
mshadow::cuda::MapPlanKernel<mshadow::sv::saveto,                   int=8,
mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu,  int=2,  float>,  float>,
mshadow::expr::Plan<mshadow::expr::ReduceWithAxisExp<mshadow::red::maximu
m,  mshadow::Tensor<mshadow::gpu,  int=3,  float>,  float,  int=3,  bool=1,
int=2>, float>>(mshadow::gpu, unsigned int, mshadow::Shape<int=2>, int=2)
  0.01%  9.7920us        1  9.7920us  9.7920us  9.7920us  [CUDA memcpy DtoH]
```

```
==311== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 46.99%    1.89525s                18    105.29ms    22.089us    947.42ms
cudaStreamCreateWithFlags
 28.30%  1.14135s       10  114.14ms     985ns  323.79ms  cudaFree
 20.87%  841.66ms       24  35.069ms  238.59us  834.55ms  cudaMemGetInfo
  3.20%    129.08ms                25    5.1632ms    5.4510us    83.800ms
cudaStreamSynchronize
  0.30%  12.147ms        8  1.5184ms  7.8030us  4.3403ms  cudaMemcpy2DAsync
  0.16%  6.6128ms       42  157.45us  12.133us  1.1190ms  cudaMalloc
  0.06%  2.3774ms        4  594.35us  25.282us  2.2295ms  cudaStreamCreate
  0.03%  1.3671ms        4  341.77us  339.61us  344.71us  cuDeviceTotalMem
  0.03%    1.1004ms              114    9.6520us          683ns    477.44us
cudaEventCreateWithFlags
  0.02%    838.06us              352    2.3800us          248ns    63.434us
cuDeviceGetAttribute
  0.01%  462.77us       23  20.120us  10.799us  89.038us  cudaLaunch
  0.01%  352.34us        6  58.723us  27.564us  123.35us  cudaMemcpy
  0.00%  112.09us        4  28.022us  18.544us  42.298us  cuDeviceGetName
  0.00%    88.915us              110          808ns          453ns    2.3590us
cudaDeviceGetAttribute
  0.00%  78.101us       32  2.4400us     666ns  7.0210us  cudaSetDevice
  0.00%    56.707us                 2    28.353us    23.198us    33.509us
cudaStreamCreateWithPriority
0.00%  54.966us      147     373ns     253ns     978ns  cudaSetupArgument
  0.00%  28.173us       10  2.8170us     873ns  8.3190us  cudaGetDevice
  0.00%  24.061us       23  1.0460us     462ns  2.1600us  cudaConfigureCall
  0.00%  9.7640us        1  9.7640us  9.7640us  9.7640us  cudaBindTexture
  0.00%  7.4550us       16     465ns     335ns     820ns  cudaPeekAtLastError
  0.00%    6.1250us                 1    6.1250us    6.1250us    6.1250us
cudaStreamGetPriority
  0.00%  5.0170us        6     836ns     259ns  1.6570us  cuDeviceGetCount
  0.00%    3.9130us                 2    1.9560us    1.4330us    2.4800us
cudaStreamWaitEvent
  0.00%    3.7470us                 2    1.8730us    1.6780us    2.0690us
cudaDeviceGetStreamPriorityRange
  0.00%  3.5000us        2  1.7500us  1.1870us  2.3130us  cudaEventRecord
  0.00%  3.2560us        6     542ns     372ns     769ns  cuDeviceGet
  0.00%  3.1920us        6     532ns     317ns     701ns  cudaGetLastError
  0.00%  2.7540us        3     918ns     883ns     984ns  cuInit
  0.00%  1.9650us        3     655ns     614ns     677ns  cuDriverGetVersion
  0.00%  1.5660us        1  1.5660us  1.5660us  1.5660us  cudaUnbindTexture
  0.00%    936ns        1     936ns     936ns     936ns  cudaGetDeviceCount
```

The most time-consuming kernel is *cudnn::detail::implicit_convolve_sgemm*, which consumed 50.477ms, which occupied 37.04% of time of kernel calls.

Milestone 2

CPU Forward Implementation

In this stage, serial forward convolution was implemented.

File ece408_src/new-forward.h was modified to implement the serial forward convolution code. File rai_build.yml was also modified by changing the execution of python file into "python m2.1.py".

Both model ece408-low and ece408-high was tried with the full dataset – size of 10000. The result was shown below.

```
* Running python m2.1.py ece408-high 10000
 New Inference
 Loading fashion-mnist data... done
 Loading model... done
 Op Time: 0.115659
 Correctness: 0.8562 Model: ece408-high

* Running python m2.1.py ece408-low 10000
 New Inference
 Loading fashion-mnist data... done
 Loading model... done
 Op Time: 0.115829
 Correctness: 0.629 Model: ece408-low
```

Milestone 3

GPU Forward Implementation

In this stage, parallel forward convolution was implemented.

File ece408_src/new-forward.cuh was modified to implement the parallel forward convolution code. File rai_build.yml was also modified by changing the execution of python file into "python m3.1.py".

Our parallel implementation parallelized the batch size B, the number of output feature map M, the height and width of each feature map H, W. We used a 24*24 tile for each thread block to compute the convolution result for elements in this tile. Thus the block dimension we set was TILE_WIDTH*TILE_WIDTH*1, and the grid dimension is Batch_Size*Num_of_Feature_Maps*Num_of_Tiles.

Both model ece408-low and ece408-high was tried with the full dataset – size of 10000 under the above implementation. The result was shown below.

```
NVPROF is profiling process 311, command: python m3.1.py ece408-
low 10000
Loading model... done
Op Time: 0.449144
Correctness: 0.629 Model: ece408-low
==311== Profiling application: python m3.1.py ece408-low 10000

NVPROF is profiling process 312, command: python m3.1.py ece408-
high 10000
Loading model... done
Op Time: 0.449597
Correctness: 0.8562 Model: ece408-high
==312== Profiling application: python m3.1.py ece408-high 10000
```