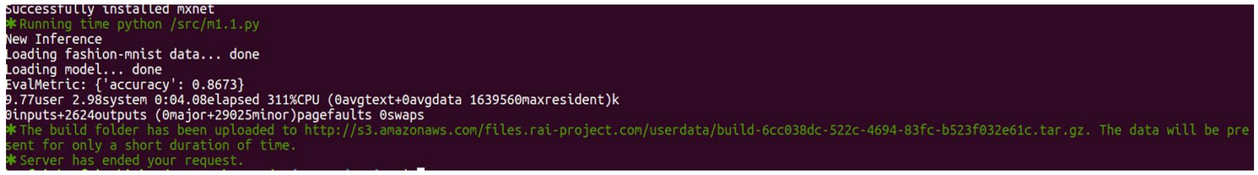
ECE408 Report 1

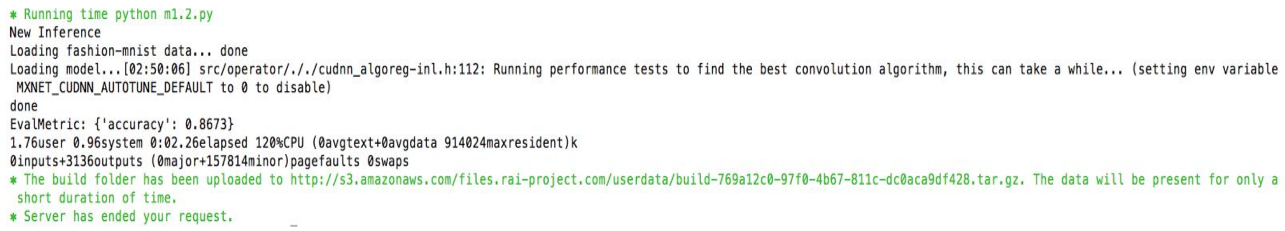
This report we summarize our work on Milestone 1 and 2. In Milestone 1, we ran MXNet baseline forward on both CPU and GPU, and generated a profile results of GPU forward pass. In Milestone2, we implemented a CPU convolution pass in MXNet. All the work is done with the help of RAI, which upload the project folder to AWS, executes steps in build file, and generates the results. The deliverable results are included as follows.

Milestone 1.1 Run the Baseline Forward Pass



```
Successfully installed mxnet
✱Running time python /src/m1.1.py
New Inference
Loading fashion-mnist data... done
Loading model... done
EvalMetric: {'accuracy': 0.8673}
9.77user 2.98system 0:04.08elapsed 311%CPU (0avgtext+0avgdata 1639560maxresident)k
0inputs+2624outputs (0major+29025minor)pagefaults 0swaps
✱The build folder has been uploaded to http://s3.amazonaws.com/files.rai-project.com/userdata/build-6cc038dc-522c-4694-83fc-b523f032e61c.tar.gz. The data will be pre
sent for only a short duration of time.
✱Server has ended your request.
```

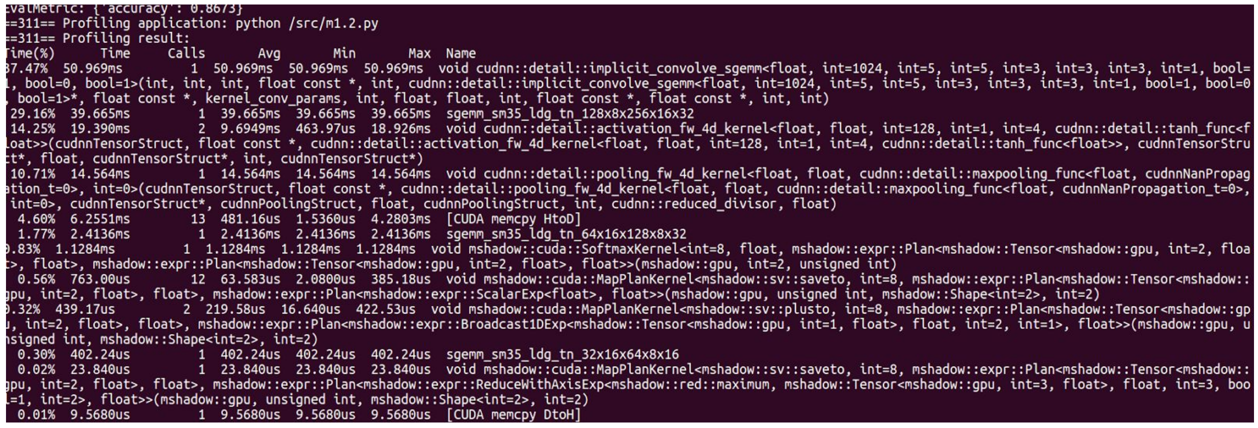
Milestone 1.2 Run the Baseline GPU Implementation



```
* Running time python m1.2.py
New Inference
Loading fashion-mnist data... done
Loading model...[02:50:06] 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}
1.76user 0.96system 0:02.26elapsed 120%CPU (0avgtext+0avgdata 914024maxresident)k
0inputs+3136outputs (0major+157814minor)pagefaults 0swaps
* The build folder has been uploaded to http://s3.amazonaws.com/files.rai-project.com/userdata/build-769a12c0-97f0-4b67-811c-dc0aca9df428.tar.gz. The data will be present for only a
 short duration of time.
* Server has ended your request.
```

It is confirmed that the accuracy of milestone 1.1 and 1.2 matches the one online. The time elapse of GPU(0:02.26) is faster than the CPU(0:04.08).

Milestone 1.3 Generate NVPROF profile

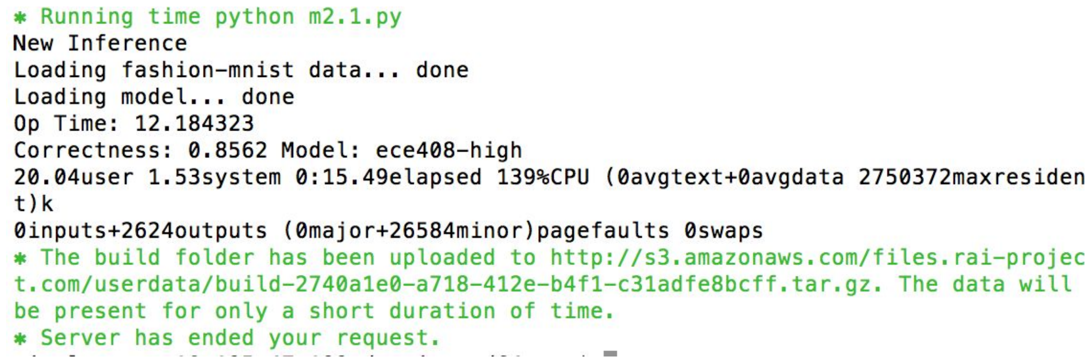


```
EvalMetric: {'accuracy': 0.8673}
==311== Profiling application: python /src/m1.2.py
==311== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
37.47%  50.969ms         1  50.969ms  50.969ms  50.969ms  void cudnn::detail::implicit_convolve_sgemm<float, int=1024, int=5, 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)
 29.16%  39.665ms         1  39.665ms  39.665ms  39.665ms  sgemm_sm35_ldg_tn_128x8x256x16x32
 14.25%  19.390ms         2  9.6949ms  463.97us  18.926ms  void cudnn::detail::activation_fw_4d_kernel<float, float, int=128, int=1, int=4, cudnn::detail::tanh_func<f
loat>>(cudnnTensorStruct, float const *, cudnn::detail::activation_fw_4d_kernel<float, float, int=128, int=1, int=4, cudnn::detail::tanh_func<float>>, cudnnTensorStru
ct*, float, cudnnTensorStruct*, int, cudnnTensorStruct*)
 10.71%  14.564ms         1  14.564ms  14.564ms  14.564ms  void cudnn::detail::pooling_fw_4d_kernel<float, float, cudnn::detail::maxpooling_func<float, cudnnNanPropag
ation_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)
  4.60%  6.2551ms        13  481.16us  1.5360us  4.2803ms  [CUDA memcpy HtoD]
  1.77%  2.4136ms         1  2.4136ms  2.4136ms  2.4136ms  sgemm_sm35_ldg_tn_64x16x128x8x32
0.83%  1.1284ms         1  1.1284ms  1.1284ms  1.1284ms  void mshadow::cuda::SoftmaxKernel<int=8, float, mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu, int=2, floa
t>, float>, mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu, int=2, float>, float>>(mshadow::gpu, int=2, unsigned int)
  0.56%  763.00us        12  63.583us  2.0800us  385.18us  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%  439.17us         2  219.58us  16.640us  422.53us  void mshadow::cuda::MapPlanKernel<mshadow::sv::plusto, int=8, mshadow::expr::Plan<mshadow::Tensor<mshadow::gp
u, 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, u
nsigned int, mshadow::Shape<int=2>, int=2)
  0.30%  402.24us         1  402.24us  402.24us  402.24us  sgemm_sm35_ldg_tn_32x16x64x8x16
  0.02%  23.840us         1  23.840us  23.840us  23.840us  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::maximum, mshadow::Tensor<mshadow::gpu, int=3, float>, float, int=3, boo
l=1, int=2>, float>>(mshadow::gpu, unsigned int, mshadow::Shape<int=2>, int=2)
  0.01%  9.5680us         1  9.5680us  9.5680us  9.5680us  [CUDA memcpy DtoH]
```

NVPROF is used to evaluate how effective the optimizations are.The kernels time is sorted and listed in the graph above in descending way. The most time consuming kernels are implicit_convolve_sgemm,void cudnn::detail::activation_fw_4d_kernel, and void cudnn::detail::pooling_fw_4d_kernel

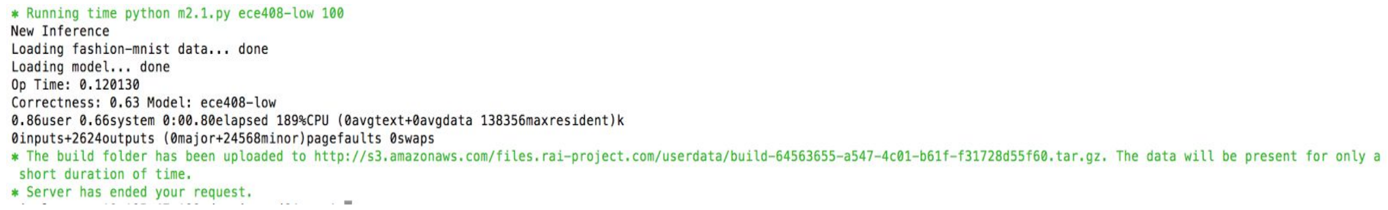
Milestone 2

A new CPU convolution layer is implemented. The correctness and time is attached in graph as follow. The default model is ece408-high with 10000 batch size.



```
* Running time python m2.1.py
New Inference
Loading fashion-mnist data... done
Loading model... done
Op Time: 12.184323
Correctness: 0.8562 Model: ece408-high
20.04user 1.53system 0:15.49elapsed 139%CPU (0avgtext+0avgdata 2750372maxresiden
t)k
0inputs+2624outputs (0major+26584minor)pagefaults 0swaps
* The build folder has been uploaded to http://s3.amazonaws.com/files.rai-projec
t.com/userdata/build-2740a1e0-a718-412e-b4f1-c31adfe8bcff.tar.gz. The data will
be present for only a short duration of time.
* Server has ended your request.
```

When we changed this to ece408-low with 100 batch size, the accuracy became lower, and time elapse decreased.

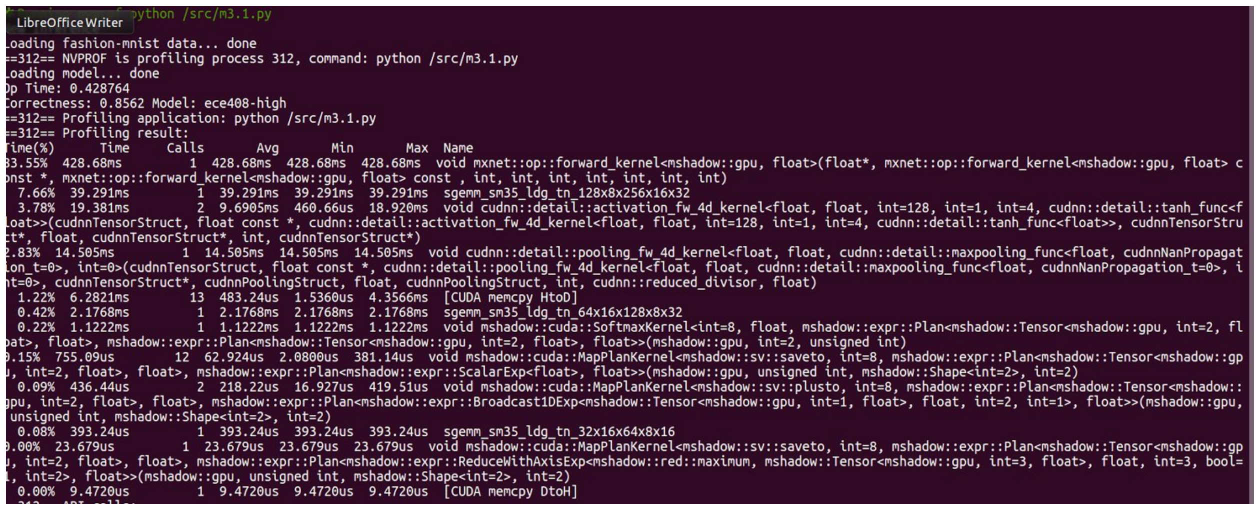


```
* Running time python m2.1.py ece408-low 100
New Inference
Loading fashion-mnist data... done
Loading model... done
Op Time: 0.120130
Correctness: 0.63 Model: ece408-low
0.86user 0.66system 0:00.80elapsed 189%CPU (0avgtext+0avgdata 138356maxresident)k
0inputs+2624outputs (0major+24568minor)pagefaults 0swaps
* The build folder has been uploaded to http://s3.amazonaws.com/files.rai-project.com/userdata/build-64563655-a547-4c01-b61f-f31728d55f60.tar.gz. The data will be present for only a
 short duration of time.
* Server has ended your request.
```

Milestone 3.1 GPU forward implementation

A GPU forward convolution is implemented with accuracy of 0.8562 and time of 0.4 seconds. Each pixel of each output feature map for each batch is computed in parallel. One point needs to be noted is that we tuned the TILE WIDTH and it shows that 24 gives the best runtime.

Milestone 3.2 GPU Profile



```
                    python /src/m3.1.py
oading fashion-mnist data... done
=312== NVPROF is profiling process 312, command: python /src/m3.1.py
oading model... done
p Time: 0.428764
orrectness: 0.8562 Model: ece408-high
=312== Profiling application: python /src/m3.1.py
=312== Profiling result:
ime(%)      Time     Calls       Avg       Min       Max  Name
3.55%  428.68ms         1  428.68ms  428.68ms  428.68ms  void mxnet::op::forward_kernel<mshadow::gpu, float>(float*, mxnet::op::forward_kernel<mshadow::gpu, float> c
nst *, mxnet::op::forward_kernel<mshadow::gpu, float> const , int, int, int, int, int, int, int)
 7.66%  39.291ms         1  39.291ms  39.291ms  39.291ms  sgemm_sm35_ldg_tn_128x8x256x16x32
 3.78%  19.381ms         2  9.6905ms  460.66us  18.920ms  void cudnn::detail::activation_fw_4d_kernel<float, float, int=128, int=1, int=4, cudnn::detail::tanh_func<f
oat>>(cudnnTensorStruct, float const *, cudnn::detail::activation_fw_4d_kernel<float, float, int=128, int=1, int=4, cudnn::detail::tanh_func<float>>, cudnnTensorStru
t*, float, cudnnTensorStruct*, int, cudnnTensorStruct*)
.83%  14.505ms         1  14.505ms  14.505ms  14.505ms  void cudnn::detail::pooling_fw_4d_kernel<float, float, cudnn::detail::maxpooling_func<float, cudnnNanPropagat
ion_t=0>, int=0>(cudnnTensorStruct, float const *, cudnn::detail::pooling_fw_4d_kernel<float, float, cudnn::detail::maxpooling_func<float, cudnnNanPropagation_t=0>, i
t=0>, cudnnTensorStruct*, cudnnPoolingStruct, float, cudnnPoolingStruct, int, cudnn::reduced_divisor, float)
 1.22%  6.2821ms        13  483.24us  1.5360us  4.3566ms  [CUDA memcpy HtoD]
 0.42%  2.1768ms         1  2.1768ms  2.1768ms  2.1768ms  sgemm_sm35_ldg_tn_64x16x128x8x32
 0.22%  1.1222ms         1  1.1222ms  1.1222ms  1.1222ms  void mshadow::cuda::SoftmaxKernel<int=8, float, mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu, int=2, fl
at>, float>, mshadow::expr::Plan<mshadow::Tensor<mshadow::gpu, int=2, float>, float>>(mshadow::gpu, int=2, unsigned int)
.15%  755.09us        12  62.924us  2.0800us  381.14us  void mshadow::cuda::MapPlanKernel<mshadow::sv::saveto, int=8, mshadow::expr::Plan<mshadow::Tensor<mshadow::gp
, int=2, float>, float>, mshadow::expr::Plan<mshadow::expr::ScalarExp<float>, float>>(mshadow::gpu, unsigned int, mshadow::Shape<int=2>, int=2)
 0.09%  436.44us         2  218.22us  16.927us  419.51us  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)
 0.08%  393.24us         1  393.24us  393.24us  393.24us  sgemm_sm35_ldg_tn_32x16x64x8x16
.00%  23.679us         1  23.679us  23.679us  23.679us  void mshadow::cuda::MapPlanKernel<mshadow::sv::saveto, int=8, mshadow::expr::Plan<mshadow::Tensor<mshadow::gp
, int=2, float>, float>, mshadow::expr::Plan<mshadow::expr::ReduceWithAxisExp<mshadow::red::maximum, mshadow::Tensor<mshadow::gpu, int=3, float>, float, int=3, bool=
, int=2>, float>>(mshadow::gpu, unsigned int, mshadow::Shape<int=2>, int=2)
 0.00%  9.4720us         1  9.4720us  9.4720us  9.4720us  [CUDA memcpy DtoH]
```

It can be seen that the accuracy is same as the one obtained in CPU code (0.8562), but it is much faster than the CPU code (0.4 seconds).

Contributions

We all worked together on the reports and codes.
Yijia Xu
Hanfei Geng
Zhaoying Du