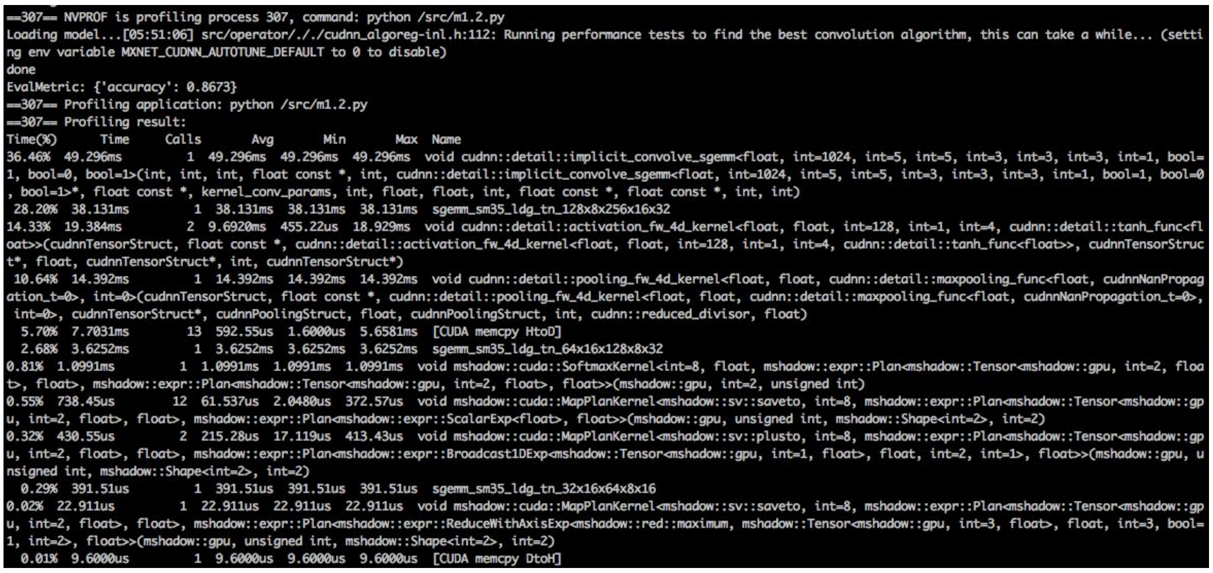
ECE408 Project Report

Team: sthBigBig
Members: Guanchen He(ghe10), Guxin Jin(gjin7), Yingyi Zhang(yingyiz2)

1.3 NVPROF Profile



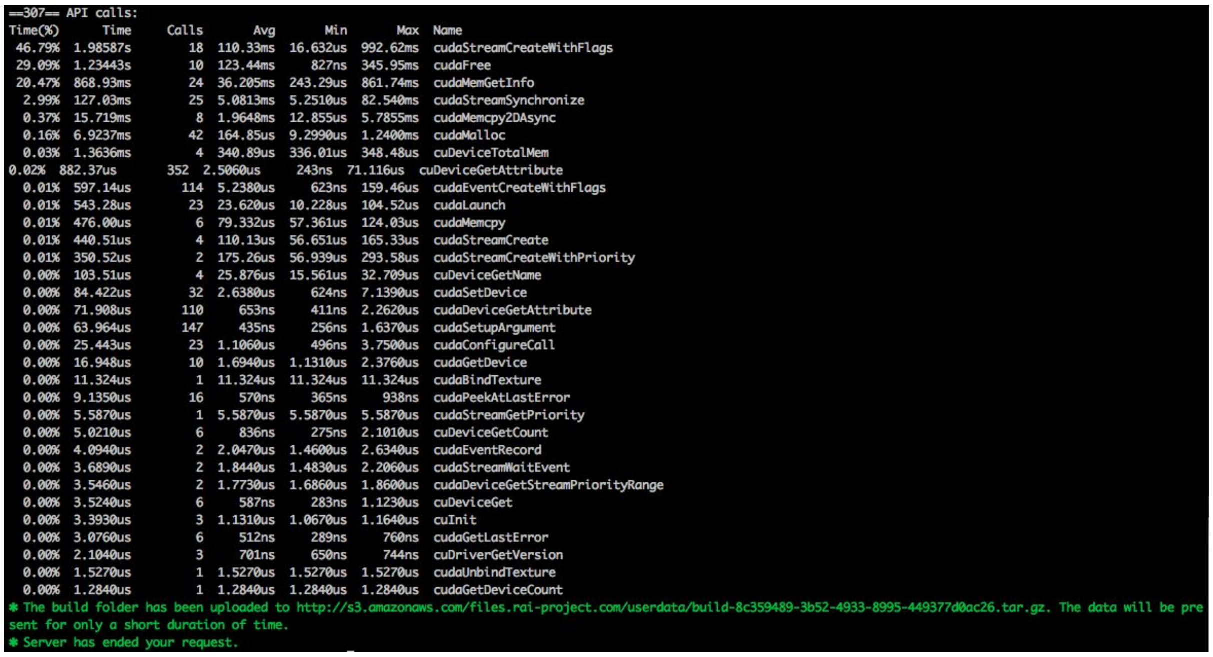
```
==307== NVPROF is profiling process 307, command: python /src/m1.2.py
Loading model...[05:51:06] src/operator/././cudnn_algoreg-inl.h:112: Running performance tests to find the best convolution algorithm, this can take a while... (setti
ng env variable MXNET_CUDNN_AUTOTUNE_DEFAULT to 0 to disable)
done
EvalMetric: {'accuracy': 0.8673}
==307== Profiling application: python /src/m1.2.py
==307== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
36.46%  49.296ms         1  49.296ms  49.296ms  49.296ms  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)
 28.20%  38.131ms         1  38.131ms  38.131ms  38.131ms  sgemm_sm35_ldg_tn_128x8x256x16x32
14.33%  19.384ms         2  9.6920ms  455.22us  18.929ms  void cudnn::detail::activation_fw_4d_kernel<float, float, int=128, int=1, int=4, cudnn::detail::tanh_func<fl
oat>>(cudnnTensorStruct, float const *, cudnn::detail::activation_fw_4d_kernel<float, float, int=128, int=1, int=4, cudnn::detail::tanh_func<float>>, cudnnTensorStruc
t*, float, cudnnTensorStruct*, int, cudnnTensorStruct*)
 10.64%  14.392ms         1  14.392ms  14.392ms  14.392ms  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)
  5.70%  7.7031ms        13  592.55us  1.6000us  5.6581ms  [CUDA memcpy HtoD]
  2.68%  3.6252ms         1  3.6252ms  3.6252ms  3.6252ms  sgemm_sm35_ldg_tn_64x16x128x8x32
0.81%  1.0991ms         1  1.0991ms  1.0991ms  1.0991ms  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)
0.55%  738.45us        12  61.537us  2.0480us  372.57us  void mshadow::cuda::MapPlanKernel<mshadow::sv::saveto, int=8, mshadow::expr::Plan<mshadow::Tensor<mshadow::gp
u, int=2, float>, float>, mshadow::expr::Plan<mshadow::expr::ScalarExp<float>, float>>(mshadow::gpu, unsigned int, mshadow::Shape<int=2>, int=2)
0.32%  430.55us         2  215.28us  17.119us  413.43us  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)
  0.29%  391.51us         1  391.51us  391.51us  391.51us  sgemm_sm35_ldg_tn_32x16x64x8x16
0.02%  22.911us         1  22.911us  22.911us  22.911us  void mshadow::cuda::MapPlanKernel<mshadow::sv::saveto, int=8, mshadow::expr::Plan<mshadow::Tensor<mshadow::gp
u, int=2, float>, float>, mshadow::expr::Plan<mshadow::expr::ReduceWithAxisExp<mshadow::red::maximum, 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%  9.6000us         1  9.6000us  9.6000us  9.6000us  [CUDA memcpy DtoH]
```

Figure 1 screenshot of profile application

Table 1 several time-consuming kernels

Time(%)	Time	Name
36.46%	49.296ms	implicit_convolve_sgemm
28.20%	38.131ms	sgemm_sm35_ldg_tn
14.33%	19.384ms	activation_fw_4d_kernel
10.64%	14.392ms	pooling_fw_4d_kernel
5.70%	7.7031ms	cuda memcpy HtoD
2.68%	3.6252ms	sgemm_sm35_ldg_tn

From the table we find that the forward activation and pooling part could be optimized by leveraging parallel algorithm or techniques discussed in this lecture.



```
==307== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 46.79%  1.98587s        18  110.33ms  16.632us  992.62ms  cudaStreamCreateWithFlags
 29.09%  1.23443s        10  123.44ms     827ns  345.95ms  cudaFree
 20.47%  868.93ms        24  36.205ms  243.29us  861.74ms  cudaMemGetInfo
  2.99%  127.03ms        25  5.0813ms  5.2510us  82.540ms  cudaStreamSynchronize
  0.37%  15.719ms         8  1.9648ms  12.855us  5.7855ms  cudaMemcpy2DAsync
  0.16%  6.9237ms        42  164.85us  9.2990us  1.2400ms  cudaMalloc
  0.03%  1.3636ms         4  340.89us  336.01us  348.48us  cuDeviceTotalMem
0.02%  882.37us       352  2.5060us     243ns  71.116us  cuDeviceGetAttribute
  0.01%  597.14us       114  5.2380us     623ns  159.46us  cudaEventCreateWithFlags
  0.01%  543.28us        23  23.620us  10.228us  104.52us  cudaLaunch
  0.01%  476.00us         6  79.332us  57.361us  124.03us  cudaMemcpy
  0.01%  440.51us         4  110.13us  56.651us  165.33us  cudaStreamCreate
  0.01%  350.52us         2  175.26us  56.939us  293.58us  cudaStreamCreateWithPriority
  0.00%  103.51us         4  25.876us  15.561us  32.709us  cuDeviceGetName
  0.00%  84.422us        32  2.6380us     624ns  7.1390us  cudaSetDevice
  0.00%  71.908us       110     653ns     411ns  2.2620us  cudaDeviceGetAttribute
  0.00%  63.964us       147     435ns     256ns  1.6370us  cudaSetupArgument
  0.00%  25.443us        23  1.1060us     496ns  3.7500us  cudaConfigureCall
  0.00%  16.948us        10  1.6940us  1.1310us  2.3760us  cudaGetDevice
  0.00%  11.324us         1  11.324us  11.324us  11.324us  cudaBindTexture
  0.00%  9.1350us        16     570ns     365ns     938ns  cudaPeekAtLastError
  0.00%  5.5870us         1  5.5870us  5.5870us  5.5870us  cudaStreamGetPriority
  0.00%  5.0210us         6     836ns     275ns  2.1010us  cuDeviceGetCount
  0.00%  4.0940us         2  2.0470us  1.4600us  2.6340us  cudaEventRecord
  0.00%  3.6890us         2  1.8440us  1.4830us  2.2060us  cudaStreamWaitEvent
  0.00%  3.5460us         2  1.7730us  1.6860us  1.8600us  cudaDeviceGetStreamPriorityRange
  0.00%  3.5240us         6     587ns     283ns  1.1230us  cuDeviceGet
  0.00%  3.3930us         3  1.1310us  1.0670us  1.1640us  cuInit
  0.00%  3.0760us         6     512ns     289ns     760ns  cudaGetLastError
  0.00%  2.1040us         3     701ns     650ns     744ns  cuDriverGetVersion
  0.00%  1.5270us         1  1.5270us  1.5270us  1.5270us  cudaUnbindTexture
  0.00%  1.2840us         1  1.2840us  1.2840us  1.2840us  cudaGetDeviceCount
* The build folder has been uploaded to http://s3.amazonaws.com/files.rai-project.com/userdata/build-8c359489-3b52-4933-8995-449377d0ac26.tar.gz. The data will be pre
sent for only a short duration of time.
* Server has ended your request.
```

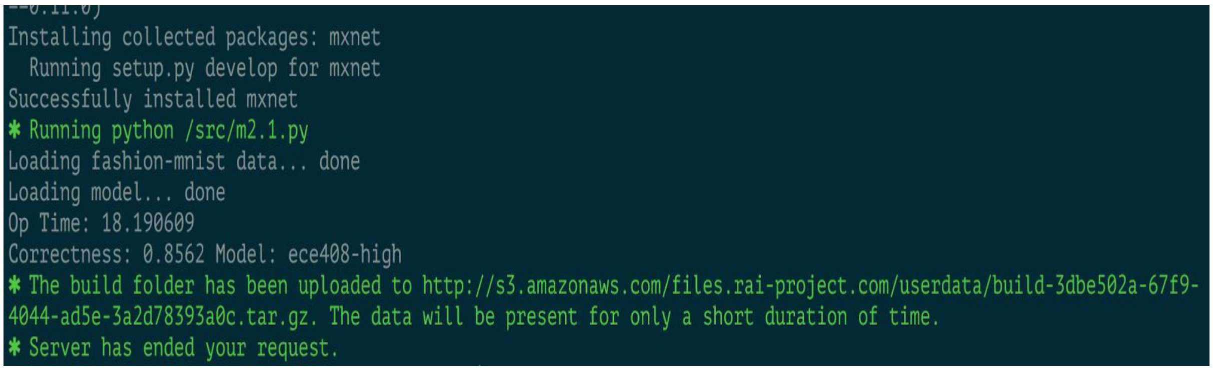
Figure 2 screenshot of API calls.

Table 2 some time-consuming API calls

Time(%)	Time	Name
46.79%	1.9858s	cudaStreamCreateWithFlag
29.09%	1.2344s	cudaFree
20.47%	868.93ms	cudaMemGetInfo
2.99%	127.03ms	cudaStreamSynchronize

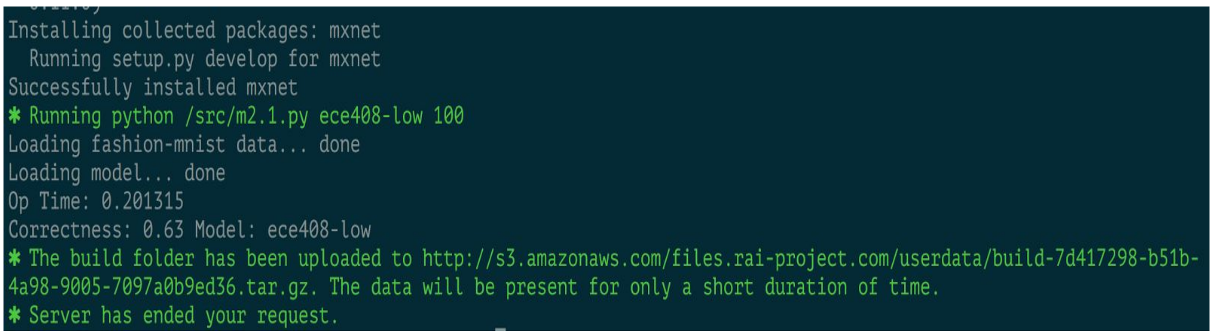
2.1 Simple CPU implementation

In this step, we implemented a CPU convolutional kernel. We flowed the forward convolution described in Chapter 16 of the textbook. The implementation is a for-loop that loop through all the computation positions and produces the convolution result. The classification results with our convolution kernel are presented in the following figures.



```
Installing collected packages: mxnet
  Running setup.py develop for mxnet
Successfully installed mxnet
* Running python /src/m2.1.py
Loading fashion-mnist data... done
Loading model... done
Op Time: 18.190609
Correctness: 0.8562 Model: ece408-high
* The build folder has been uploaded to http://s3.amazonaws.com/files.rai-project.com/userdata/build-3dbe502a-67f9-
4044-ad5e-3a2d78393a0c.tar.gz. The data will be present for only a short duration of time.
* Server has ended your request.
```

Figure 3 ece408-high model execution time and accuracy



```
Installing collected packages: mxnet
  Running setup.py develop for mxnet
Successfully installed mxnet
✱ Running python /src/m2.1.py ece408-low 100
Loading fashion-mnist data... done
Loading model... done
Op Time: 0.201315
Correctness: 0.63 Model: ece408-low
✱ The build folder has been uploaded to http://s3.amazonaws.com/files.rai-project.com/userdata/build-7d417298-b51b-
4a98-9005-7097a0b9ed36.tar.gz. The data will be present for only a short duration of time.
✱ Server has ended your request.
```

Figure 4 ece408-low model execution time and accuracy

Contribution: All the members thoroughly discussed the problem and distributed the task reasonably. Guanchen He(ghe10) wrote the CPU code; Yingyi Zhang(yingyiz2) performed the experiments; Guxin Jin(gjin7) wrote the report.