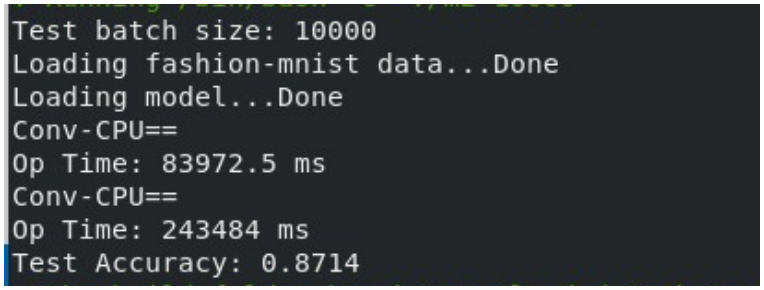
Name: Benoit Ortalo-Magne
NetId: beo2
Team name: ChampaignWithoutTheCham
School: Illinois

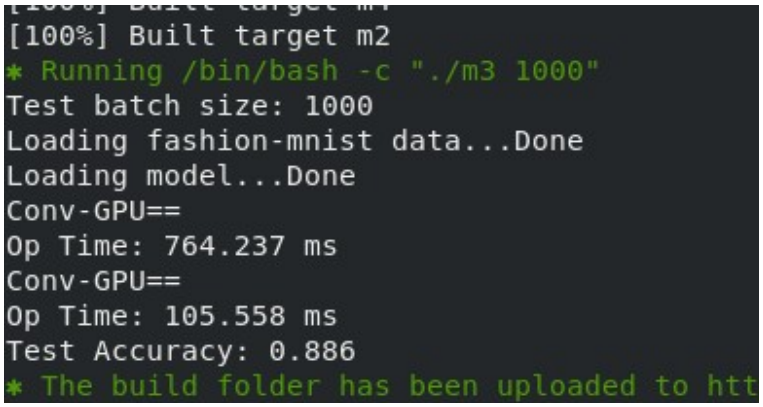
Milestone 2



Test Accuracy: .8714
Op Times: 83972.5 ms and 243484 ms
Total Time: 327456.5 ms

Milestone 3

Rai running your GPU implementation of convolution:



Nsys profiling GPU execution:

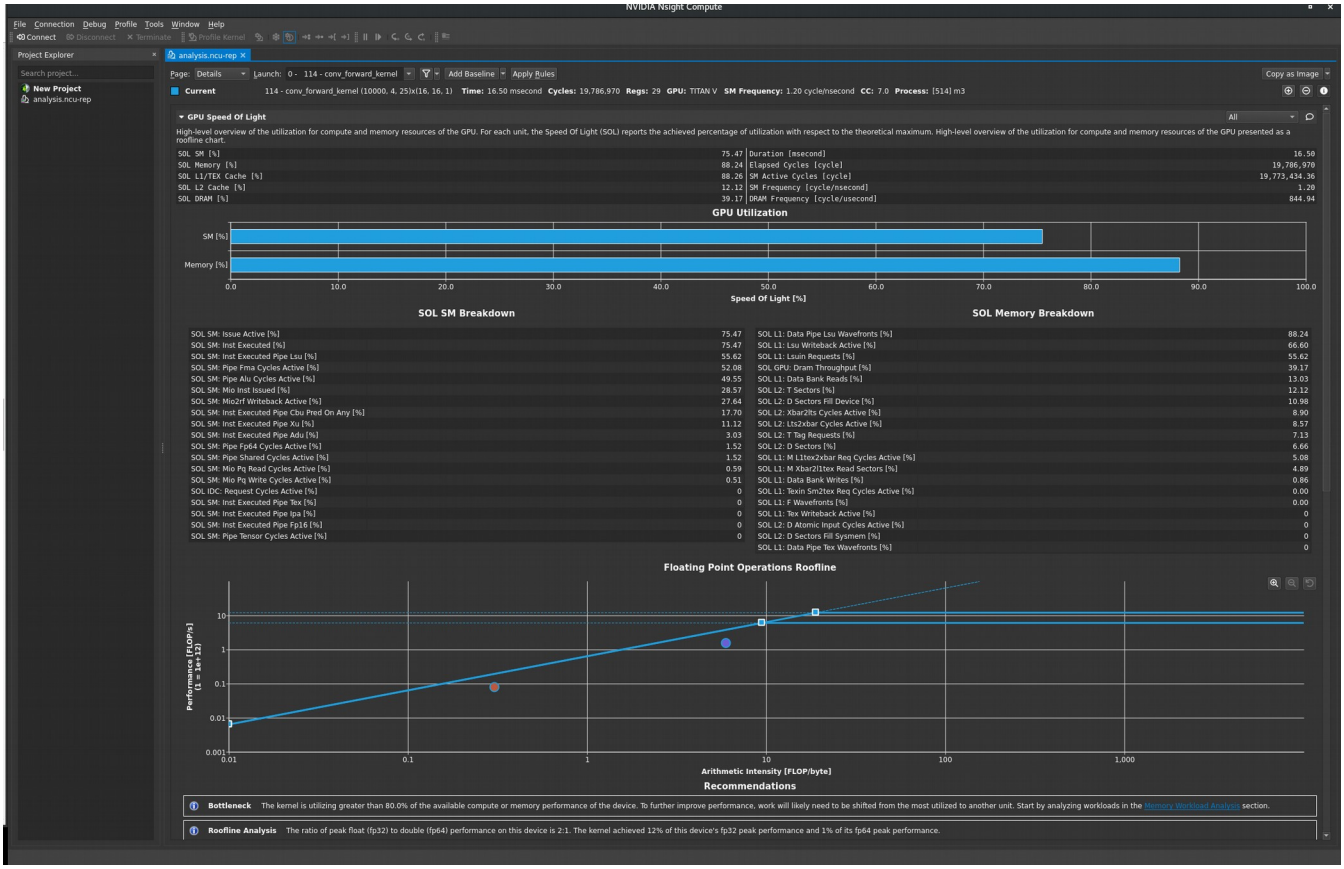
```
*
*
*
Exported successfully to
/build/report1.sqlite
Generating CUDA API Statistics...
CUDA API Statistics (nanoseconds)

Time(%)      Total Time       Calls         Average         Minimum         Maximum  Name
-------  --------------  ----------  --------------  --------------  --------------  --------------------------------
   68.7     68960734610           6   11493455768.3          169271     66200836920  cudaMemcpy
   29.3     29404750501           6    4900791750.2           81827     29401957235  cudaMalloc
    2.0      1991866714           2     995933357.0          207078      1991659636  cudaLaunchKernel
    0.0         2451533           6        408588.8           74779          958385  cudaFree
Generating CUDA Kernel Statistics...

Generating CUDA Memory Operation Statistics...
CUDA Kernel Statistics (nanoseconds)

Time(%)      Total Time   Instances         Average         Minimum         Maximum  Name
-------  --------------  ----------  --------------  --------------  --------------  --------------------------------
  100.0     61475508550           2   30737754275.0      2096648400     59378860150  conv_forward_kernel


CUDA Memory Operation Statistics (nanoseconds)

Time(%)      Total Time  Operations         Average         Minimum         Maximum  Name
-------  --------------  ----------  --------------  --------------  --------------  --------------------------------
   89.9       957054782           2     478527391.0       404156011       552898771  [CUDA memcpy DtoH]
   10.1       107626847           4      26906711.8            1216        65813173  [CUDA memcpy HtoD]


CUDA Memory Operation Statistics (KiB)

            Total      Operations            Average            Minimum            Maximum  Name
----------------  --------------  -----------------  -----------------  -----------------  --------------------------
        1722500.0               2           861250.0         722500.000          1000000.0  [CUDA memcpy DtoH]
         538919.0               4           134729.0              0.766           288906.0  [CUDA memcpy HtoD]
Generating Operating System Runtime API Statistics...
Operating System Runtime API Statistics (nanoseconds)

Time(%)      Total Time       Calls         Average         Minimum         Maximum  Name
-------  --------------  ----------  --------------  --------------  --------------  --------------------------------
   33.4    198343708192        1995      99420405.1           59553       101193117  poll
   33.1    196417907592        1978      99301267.7           52031       100731694  sem_timedwait
   22.0    130534202112         261     500131042.6       500071491       500374049  pthread_cond_timedwait
   11.4     67555247428           2   33777623714.0     25375289661     42179957767  pthread_cond_wait
    0.0       142269536         949        149915.2            1098        16074777  ioctl
    0.0        21277872        9072          2345.4            1276            8958  read
    0.0         2694592          96         28068.7            1557          906169  mmap
    0.0          648746         101          6423.2            2228           18920  open64
    0.0          239660           5         47932.0           36760           59883  pthread_create
    0.0           91122          19          4795.9            1104           15418  munmap
    0.0           77592          24          3233.0            1002           12070  fopen
    0.0           75515           3         25171.7           12384           50438  fgets
    0.0           65680          15          4378.7            2089            8012  write
    0.0           52791           3         17597.0            2879           35121  fopen64
    0.0           42985           8          5373.1            1209            9092  fflush
    0.0           41121           1         41121.0           41121           41121  pthread_mutex_lock
    0.0           29269           5          5853.8            3732            8268  open
    0.0           17205          10          1720.5            1005            3858  fclose
    0.0           16714           3          5571.3            5357            5821  pipe2
    0.0           10378           2          5189.0            4757            5621  pthread_cond_signal
    0.0           10265           2          5132.5            3980            6285  socket
    0.0            6573           1          6573.0            6573            6573  connect
0.0            5693          1          5693.0            5693            5693  fwrite
    0.0            3310           3          1103.3            1016            1175  fcntl
    0.0            3020           1          3020.0            3020            3020  bind
Generating NVTX Push-Pop Range Statistics...
NVTX Push-Pop Range Statistics (nanoseconds)
★ The build folder has been uploaded to http://s3.amazonaws.com/files.rai-project.com/userdata/build-5f90d515c56e891b1
```

Kernels that consume more than 90% of program time: conv_forward_kernel

CUDA API calls that consume more than 90% of program time: cudaMemcpy, cudaMalloc, cudaLaunchKernel, cudaFree

Difference between kernels and API calls: API calls are used to set up memory and variables for kernels to run on.

Screenshot of GPU SOL utilization:



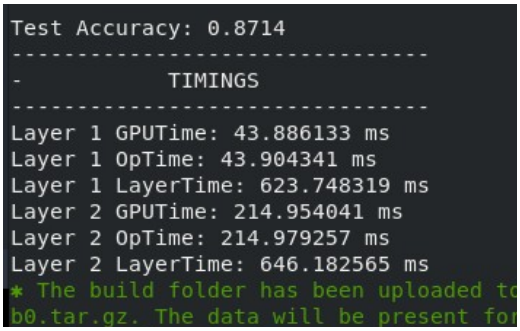
Milestone 4

How you identified the optimization opportunity: kernel memory is made up of constant values so they can be stored in constant memory to reduce load times while running the kernel

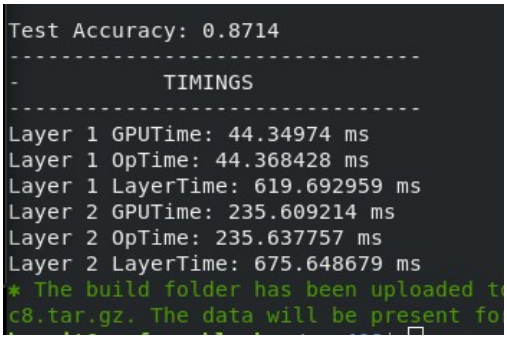
Why the approach is fruitful: It allows for faster load times when performing calculations using kernel memory

The effect: Overall, the time taken is decreased. This implementation suffers a bit in the Layer 1 layer time but makes up for it everywhere else.

With constant memory



Without constant memory



Work was divided up by optimization, each of us wrote one.