|  |  |
| --- | --- |
| **Name:** | *Yiming Li* |
| **NetID:** | *yiming22* |
| **Section:** | *AL1* |

**ECE 408/CS483 Milestone 2 Report**

|  |
| --- |
| 1. Show output of rai running Mini-DNN on the basic GPU convolution implementation for batch size of 1k images. This can either be a screen capture or a text copy of the running output. Please do not show the build output. (The running output should be everything including and after the line "*Loading fashion-mnist data...Done*"). |
| *<output here>*  *Loading fashion-mnist data...Done*  *Loading model...Done*  *Conv-GPU==*  *Layer Time: 68.8763 ms*  *Op Time: 1.63944 ms*  *Conv-GPU==*  *Layer Time: 51.8532 ms*  *Op Time: 6.2645 ms*  *Test Accuracy: 0.886*  *real 0m9.660s*  *user 0m9.269s*  *sys 0m0.328s*  *✱ The build folder has been uploaded to http://s3.amazonaws.com/files.rai-project.com/userdata/build-6184c17df5b88942145451fb.tar.gz. The data will be present for only a short duration of time.* |
| 1. For the basic GPU implementation, list Op Times, whole program execution time, and accuracy for batch size of 100, 1k, and 10k images. |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Batch Size | Op Time 1 | Op Time 2 | Total Execution Time | Accuracy | | 100 | *0.175098 ms* | *0.632573 ms* | *0m1.190s* | *0.86* | | 1000 | *1.63944 ms* | *6.2645 ms* | *0m9.660s* | *0.886* | | 10000 | 16.0852 ms | 62.7071 ms | 1m37.766s | 0.8714 | |
| 1. List all the kernels that collectively consumed more than 90% of the kernel time and what percentage of the kernel time each kernel did consume (start with the kernel that consumed the most time, then list the next kernel, until you reach 90% or more). |
| *<answer here>*  *conv\_forward\_kernel (100.0% Time)* |
| 1. List all the CUDA API calls that collectively consumed more than 90% of the API time and what percentage of the API time each call did consume (start with the API call that consumed the most time, then list the next call, until you reach 90% or more). |
| *<answer here>*  *cudaMemcpy (76.6% Time)*  *cudaMalloc (15.9% Time)* |
| 1. Explain the difference between kernels and CUDA API calls. Please give an example in your explanation for both. |
| *<answer here>*  *Kernels are C++ functions that are executed N times in parallel by N different CUDA threads when called. A kernel is defined using the \_\_global\_\_ declaration specifier.*  *Example is the kernel "conv\_forward\_kernel()" in this milestone2 code.*  *However, API calls are calls made by the code into the CUDA driver or runtime libraries. It is only executed once like regular C++ functions.*  *Examples are "cudaMalloc()", "cudaMemcpy()", etc.* |
| 1. Show a screenshot of the GPU SOL utilization |
| *<nsight output here>*  *Launch: 1 - 122 - conv\_forward\_kernel*    Launch: 4 - 143 - conv\_forward\_kernel |