Programming Language, Assignment 6
Due: June 18, 11:59 pm

For this assignment, use the attached file (hw6.cu) as a skeleton code. You must replace the occurrence of "CHANGE" as necessary to complete the problem.

Overview

You will implement two core functions that are used in deep learning – max pooling and GEMM (general matrix-matrix multiplication). The problems below describe what you need to implement, but for more details and background information, please refer to https://en.wikipedia.org/wiki/Convolutional_neural_network#Pooling_layer and https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

Submission

Write the functions in problem 1 – 2 in the given file "*hw6.cu*" and upload it to the course homepage (under the assignment 6 menu). Make sure that your code compiles and runs correctly -- otherwise you will get zero score.

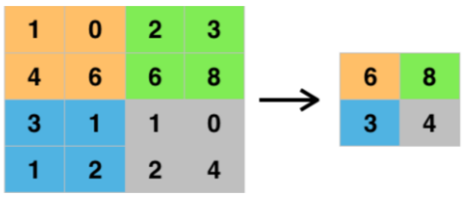
Problems

1. **Max Pooling (10 pts) (warming up)**

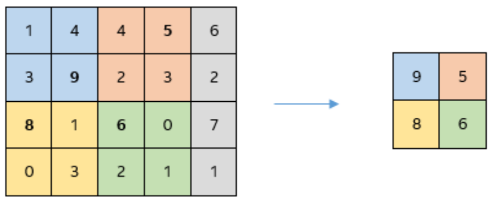
   In this problem, you need to implement the CUDA kernel "maxpool":

   void maxpool (float *input, float *output, int input_size, int filter_size);

   The above kernel performs max pooling, which is selecting the maximum values for each "grid" in the given matrix. More specifically, given the input matrix (input) and the filter size (filter_size), the input matrix is divided into a grid of submatrices of filter size as the following left matrix (the colors indicate different grids). Then in each submatrix, select the maximum value and store in the output matrix (output). For example, in the following, the matrix is 4 * 4, and the filter size is 2; hence the matrix is divided into a grid of 2*2 submatrices. Then the maximum values in each submatrix are stored in the output matrix. Assume that input matrix size is larger than the filter size. Also assume that the input matrix is square matrix (N*N) and the filter is a square.



   If there are left-overs, as the rightmost column in the following, they are ignored.



2. **Simple GEMM (40 pts)**

   You need to implement a simplified GEMM operation. GEMM, or generalized matrix multiplication, refers to the following operation:

   $Y = \alpha A \times B + \beta C$

   where A, B, C are matrices and α and β are scalar value. For this problem, assume that A, B, C are square matrices with float values (32 bit, single precision).

   Your implementation should use the tiled operation to first copy a part of input matrices into shared memory then perform the (multiplication or addition) operation as discussed in the class.

   You should consider the case that the matrix sizes are not multiple of tile size; at the same time, you should consider warp convergence for better performance. You can assume that the matrix sizes are always larger than the tile size.

   For this problem, your solution will be graded based on the performance. Your solution should (1) give correct answer and (2) run fast.