CS133 Lab 4 Report
Yanzhe Liu 504153440

Parallelization Strategy:
I used a 3-D global work group size {256, 224, 224}, in which each thread would work on 1 pixel of the middle convolutional layer. I then divide the local group size using 3-D {1, 32, 32}, so that in each work group, 1024 threads would work on 1 layer's 32x32 block pixels. I used loop tiling in convolution J step to further utilize the local buffer.

The communication overhead in this implementation mainly involves in the reading and writing local buffers from global memory.

Execution Time:
I measured the execution time (GFlop/s) on aws server. The kernel execution is about 70 GFlop/s while the overall execution time (including OpenCL setup) is around 25 GFlop/s.

If only kernel time is compared, the GPU implementation is slightly slower than the CPU implementation. I think it is because CPU implementation can have larger local memory (buffer), making the repeat memory access faster than GPU. And in CPU I explicit loop permutation to enhance locality, but in GPU it is not possible due to the structure of local work groups.

Memory Usage:
Private:
    I used 9 local variable to store local ids.
Local:
    Cin_local: 8 * 36 * 36 * 4 = 41,472 Bytes = 40.5 KB
    w_local: 8 * 5 * 5 * 4 = 800 Bytes
Global:
    Cin: 256 * 228 * 228 * 4 = 53,231,616 Bytes = 50 MB
    Weight: 256 * 256 * 5 * 5 * 4 = 6,553,600 Bytes = 6.25 MB
    Bias: 256 * 4 = 1024 Bytes = 1 KB
    Cconv: 256 * 224 * 224 * 4 = 51,380,224 Bytes = 49 MB

Challenges:
The most challenging part of this project is to figure out the right dimension for local work group. Buffer Cin is also a bit tricky since it is slightly larger than the middle layer block size.