CPE 810 Lab 03

Zirui Wen

Question 1

Convolution is a widely used operation across various fields.

- Image Processing and Computer Vision
- Audio and Signal Processing
- Scientific Computing (Stencils) and Engineering Simulations

Question 2

Each output pixel requires multiplying each mask element by the corresponding image pixel and summing them up.

$$\begin{aligned} FLOPS &\approx dimX \times dimY \times (dimK^2 + (dimK^2 - 1)) \\ &\approx 2 \times dimX \times dimY \times (dimK^2) \end{aligned}$$

Question 3

For the basic method, each thread reads each needed pixel from global memory without reuse. So the total reads from global memory are:

$$Read \approx dimX \times dimY \times (dimK^2)$$

After using the shared memory, each input element is loaded once into shared memory and then reused by many threads. The total read is:

$$Read = T \times (BLOCK\_SIZE + dimK - 1)^2$$

Question 4

We produce one output value per image pixel:

$$Write = dimX \times dimY$$

Question 5

Max

Maximum operations for threads that are completely interior in the image (i.e., their entire $dimK \times dimK$ neighborhood lies within the image) will perform the full $dimK^2$ multiplications and $dimK^2 - 1$ additions. The maximum total operations

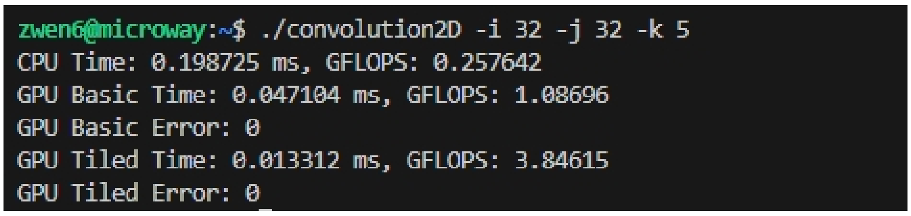
Min

For minimum operations, we think of the position (0,0). The multiple operation is $(\lfloor \frac{dimK}{2} \rfloor + 1)^2$, the adding operation is $(\lfloor \frac{dimK}{2} \rfloor + 1)^2 - 1$, the total operation is $\frac{1}{2}dimK^2$.

The average operation times are between $\frac{1}{2}dimK^2$ and $dimK^2 - 1$

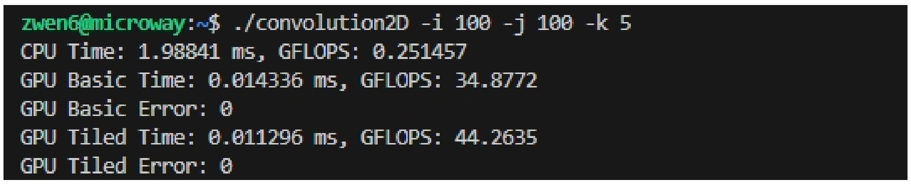
Question 6

Here are some screenshot of the results:



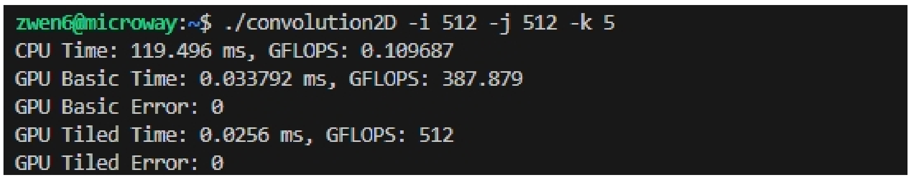
```
zwen6@microway:~$ ./convolution2D -i 32 -j 32 -k 5
CPU Time: 0.198725 ms, GFLOPS: 0.257642
GPU Basic Time: 0.047104 ms, GFLOPS: 1.08696
GPU Basic Error: 0
GPU Tiled Time: 0.013312 ms, GFLOPS: 3.84615
GPU Tiled Error: 0
```

Figure 1: 32 * 32



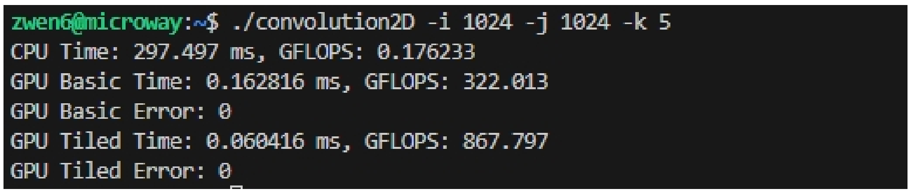
```
zwen6@microway:~$ ./convolution2D -i 100 -j 100 -k 5
CPU Time: 1.98841 ms, GFLOPS: 0.251457
GPU Basic Time: 0.014336 ms, GFLOPS: 34.8772
GPU Basic Error: 0
GPU Tiled Time: 0.011296 ms, GFLOPS: 44.2635
GPU Tiled Error: 0
```

Figure 2: 100 * 100



```
zwen6@microway:~$ ./convolution2D -i 512 -j 512 -k 5
CPU Time: 119.496 ms, GFLOPS: 0.109687
GPU Basic Time: 0.033792 ms, GFLOPS: 387.879
GPU Basic Error: 0
GPU Tiled Time: 0.0256 ms, GFLOPS: 512
GPU Tiled Error: 0
```

Figure 3: 512 * 512



```
zwen6@microway:~$ ./convolution2D -i 1024 -j 1024 -k 5
CPU Time: 297.497 ms, GFLOPS: 0.176233
GPU Basic Time: 0.162816 ms, GFLOPS: 322.013
GPU Basic Error: 0
GPU Tiled Time: 0.060416 ms, GFLOPS: 867.797
GPU Tiled Error: 0
```

Figure 4: 1024 * 1024

One important observation: For very small images, the GPU might not outperform the CPU by much because it's underutilized (the kernel launch and memory copy overhead dominate). But as soon as the image gets moderately large (e.g., 512×512 or more), the GPU's massive parallelism kicks in.

Question 7

Analysis of the 2D Convolution Kernel Code

This code utilizes shared memory to accelerate a 2D convolution operation. The shared memory block is allocated with a width defined as

$$w = \text{BLOCK\_SIZE} + \text{dimK} - 1,$$

With $\text{BLOCK\_SIZE} = 16$, each block contains 256 threads. The kernel is designed so that each thread loads two elements, resulting in a total of

$$2 \times 256 = 512$$

shared memory loads per block.

Correct Operation with a Small Mask

For a small mask, such as a 3×3 filter (i.e., $\text{dimK} = 3$):

$$\text{Required shared memory size} = (16 + 3 - 1)^2 = 18^2 = 324 \text{ elements.}$$

Since 512 elements are loaded into shared memory, all necessary data is correctly initialized, and the convolution operates as expected.

Issue with Larger Masks

When the mask size increases (for example, a 9×9 filter with $\text{dimK} = 9$):

$$\text{Required shared memory size} = (16 + 9 - 1)^2 = 24^2 = 576 \text{ elements.}$$

However, the code still performs only 512 load operations. This means that

$$576 - 512 = 64 \text{ elements}$$

in the shared memory array remain uninitialized. These uninitialized values are then used during the convolution calculation, leading to incorrect accumulation results and, consequently, inaccurate output.

The inaccuracy in the convolution result for mask sizes greater than 9 is due to an insufficient amount of data being loaded into shared memory. The loading strategy does not account for the increased number of elements required by a larger mask, resulting in uninitialized values being used in the computation.