d

Q1) How many Flops needs to be done to compute a convolution of dimension k on a image of size

n \*m?

* Our Image size is n\*m and our kernel size which would be used for convolution is k\*k
* In convolution, for a single pixel, we will do 2 times k\*k operations ie with our kernel we will do multiplication first and then addition.
* So for an image with a size of m\*n, we will do a total of (m\*n\*k\*k\*2) flops.

Q2) How much memory needs to be moved to compute a convolution of dimension k on an image of

size n\*m?

* We will bring the image of size m\*n from memory also the kernel of size k\*k.
* So we will need to fetch ((m\*n)+ (k\*k))\*4 bytes from memory.
* Also, we will write back (m\*n)\*4 bytes to memory.

Q3) Assuming the performance numbers you measured in assignment 1 and 2, how long should

computing a convolution of dimension 3 on an image of 1024x768 take?

* Considering m\*n= 1024 \* 768 and k\*k=3\*3.
* Flop time for calculation= ((m\*n\*k\*k\*2)/ 1.7\*10^12))= 8.32 usec.
* Memory time calculation= (2(m\*n)+ (k\*k))\*4)/100\*10^9 = 62.9 usec.
* Now calculating time for dimension 11.
* Flop time for calculation= ((m\*n\*k\*k\*2)/ 1.7\*10^12))= 111.9 usec.
* Memory time calculation= (2(m\*n)+ (k\*k))\*4)/100\*10^9 = 62.9 usec.

**Performance Model**

Peak flop= ((1024 \*768)/ 8.32 \* 10^-6) = 94523 ppi.

Memory flops= ((1024 \*768)/ 62.9 \* 10^-6) = 12,502 ppi.

Peak flops= ((1024 \*768)/ 111.9 \* 10^-6)= 7,027 ppi.

Memory flops= ((1024 \*768)/ 62.9 \* 10^-6)= 12,502 ppi.

Q4) Measuring and reporting performance

* N= 1024, M= 768, k= 3 , Expected- 91 gigapixels, Flop bound, Actual- 5.5 Gigapixels
* N= 1024, M= 768, k= 5 , Expected- 32 gigapixels, Flop bound, Actual- 5.1 Gigapixels
* N= 1024, M= 768, k= 7 , Expected- 16 gigapixels, Flop bound, Actual- 4.4 Gigapixels
* N= 1024, M= 768, k= 9 , Expected- 16 gigapixels, Memory bound, Actual- 1.5 Gigapixels
* N= 1024, M= 768, k= 11 , Expected- 16 gigapixels, Memory bound, Actual- 1.4 Gigapixels
* N= 1024, M= 768, k= 13 , Expected- 16 gigapixels, Memory bound, Actual- 0.9 Gigapixels
* N= 1024, M= 768, k= 15 , Expected- 16 gigapixels, Memory bound, Actual- 0.8 Gigapixels
* N= 2048, M= 2048, k= 3 , Expected- 91 gigapixels, Flop bound, Actual- 3 Gigapixels
* N= 2048, M= 2048, k= 5 , Expected- 32 gigapixels, Flop bound, Actual- 1.8 Gigapixels
* N= 2048, M= 2048, k= 7 , Expected- 16 gigapixels, Flop bound, Actual- 1.2 Gigapixels
* N= 2048, M= 2048, k= 9 , Expected- 16 gigapixels, Memory bound, Actual- 0.5 Gigapixels
* N= 2048, M= 2048, k= 11 , Expected- 16 gigapixels, Memory bound, Actual- 0.4 Gigapixels
* N= 2048, M= 2048, k= 13 , Expected- 16 gigapixels, Memory bound, Actual- 0.3 Gigapixels
* N= 2048, M= 2048, k= 15 , Expected- 16 gigapixels, Memory bound, Actual- 0.25 Gigapixels
* N= 8192, M= 8192, k= 3 , Expected- 91 gigapixels, Flop bound, Actual- 2.4 Gigapixels
* N= 8192, M= 8192, k= 5 , Expected- 32 gigapixels, Flop bound, Actual- 1.4 Gigapixels
* N= 8192, M= 8192, k= 7 , Expected- 16 gigapixels, Flop bound, Actual- 1.0 Gigapixels
* N= 8192, M= 8192, k= 9 , Expected- 16 gigapixels, Memory bound, Actual- 0.4 Gigapixels
* N= 8192, M= 8192, k= 11, Expected- 16 gigapixels, Memory bound, Actual- 0.3 Gigapixels
* N= 8192, M= 8192, k= 13, Expected- 16 gigapixels, Memory Bound, Actual- 0.26 Gigapixels
* N= 8192, M= 8192, k= 15 , Expected- 16 gigapixels, Memory bound, Actual- 0.2 Gigapixels
* N= 4194304, M= 768, k= 3 , Expected- 91 gigapixels, Flop bound, Actual- 4 Gigapixels
* N= 4194304, M= 768, k= 5 , Expected- 32 gigapixels, Flop bound, Actual- 1.6 Gigapixels
* N= 4194304, M= 768, k= 7 , Expected- 16 gigapixels, Flop bound, Actual- 1.5 Gigapixels
* N= 4194304, M= 768, k= 9 , Expected- 16 gigapixels, Memory bound, Actual- 1.2 Gigapixels
* N= 4194304, M= 768, k= 11 , Expected- 16 gigapixels, Memory bound, Actual- 0.7 Gigapixels
* N= 4194304, M= 768, k= 13, Expected- 16 gigapixels, Memory bound, Actual- 0.6 Gigapixels