Georgia Institute of Technology

ECE 8803: Hardware-Software Co-Design for Machine Learning Systems (HML)

Spring 2024

Lab 1A

Due: Friday, February 2, 2023 @ 11:59 pm EST

Created by: Abhimanyu Bambhaniya (abambhaniya3@gatech.edu)

Instructions

Please read the following instructions carefully.

- The lab is distributed in 2 parts (Each worth 10 points).
- You will need to modify the code in various files, generate output csv and submit them independently.
- Rename the zip according to the format: **LastName_FirstName_ECE_8803_HML_sp24_lab1A.zip**
- Submit the generated csv file separately. **DO NOT MODIFY IT.**
- It is encouraged for you to discuss homework problems amongst each other, but any copying is strictly prohibited and will be subject to Georgia Tech Honor Code.
- Late homework is not accepted unless arranged otherwise and in advance.
- Comment on your codes.
- For all problem, please post queries on piazza.

Lab Layout

Part A.1: Understanding various operators – 4 points.

- Calculate data movement for Relu, Add, Matmul, Conv2D.
- Calculate number of operations for Relu, Add, Matmul, Conv2D.

Part A.2: Runtime Computations - 2 points

- Compute time
- Memory time
- Roofline time

Part A.3: Building Neural Networks - 2 points.

- Alexnet
- Bert

Part A.4: Comparing the performance of NN on different HWs - 2 points.

Lab Setup

Please download the zip file and extract the contents to access the code base.
To run the code, you will need various python packages listed in requirements.txt.
Recommended Method: Create a new conda environment and run using it [Conda].

Recommended Editor : Install visual studio code and open this folder in it.

Lab details

Part A.1: Understanding various operators.

For this part, you would be modifying the code in operators.py.
The file has 4 classes for operator type. Fill the get_tensors() and get_num_ops() functions in each class.

In get_tensors(), calculate the number of elements of input and elements of output for each operator.

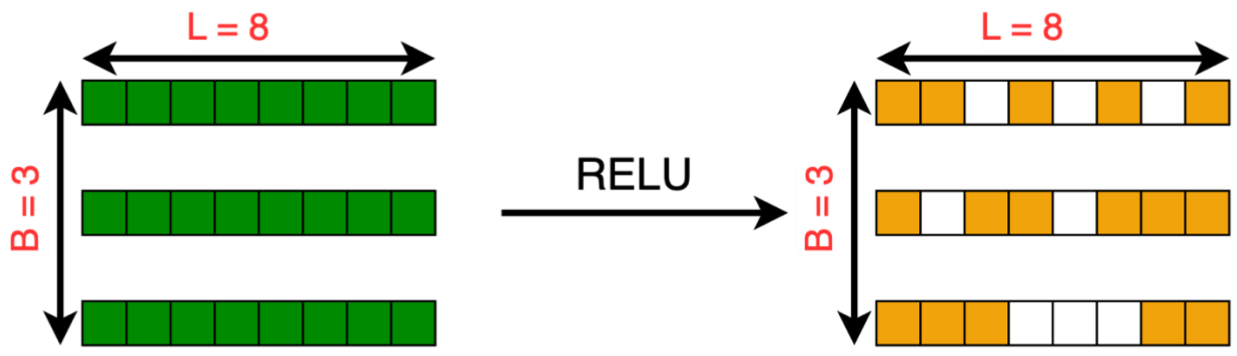
In get_num_ops(), calculate the number of operations in each operator.

Details of each of the 4 operator is explained in detail below.

A.1.i) RELU

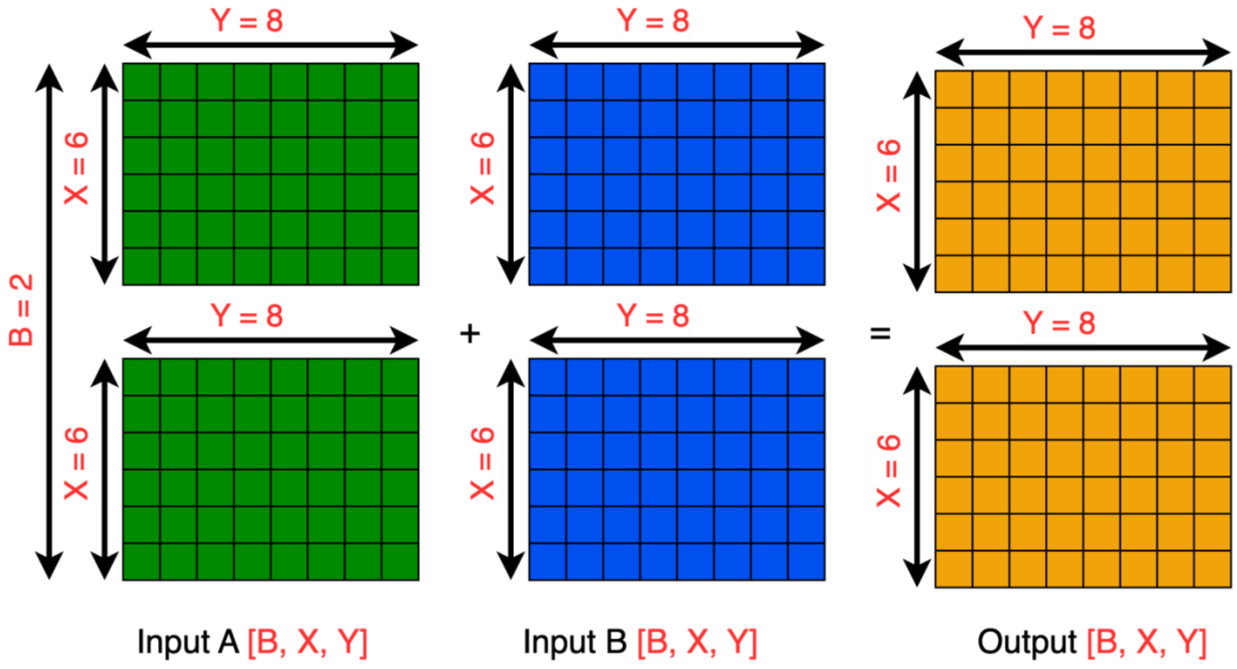
Relu(x) = x if x>0 else 0.

Assume, relu operation on each element is 1 operation.



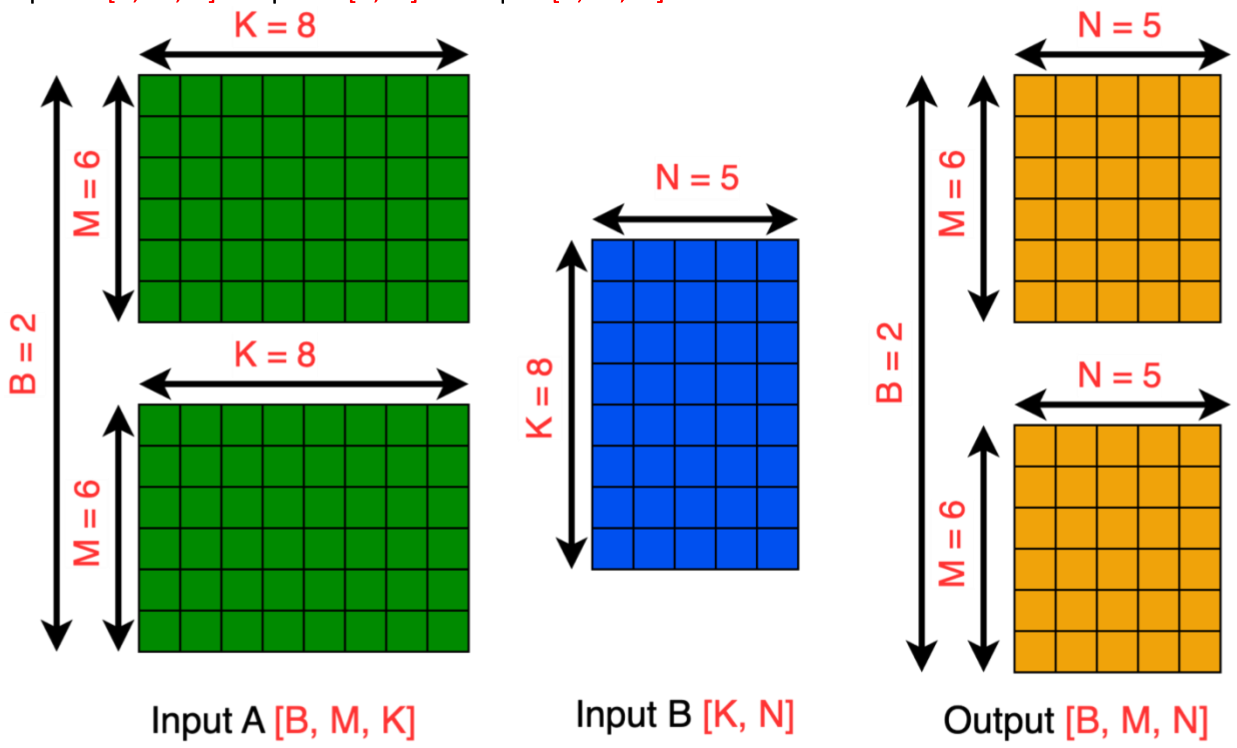
A.1.ii) Add

Input A [B, X, Y] + Input B [B, X, Y] = Output [B, X, Y]



A.1.iii) Matrix Multiplication

Input A [B, M, K] x Input B [K, N] = Output [B, M, N]

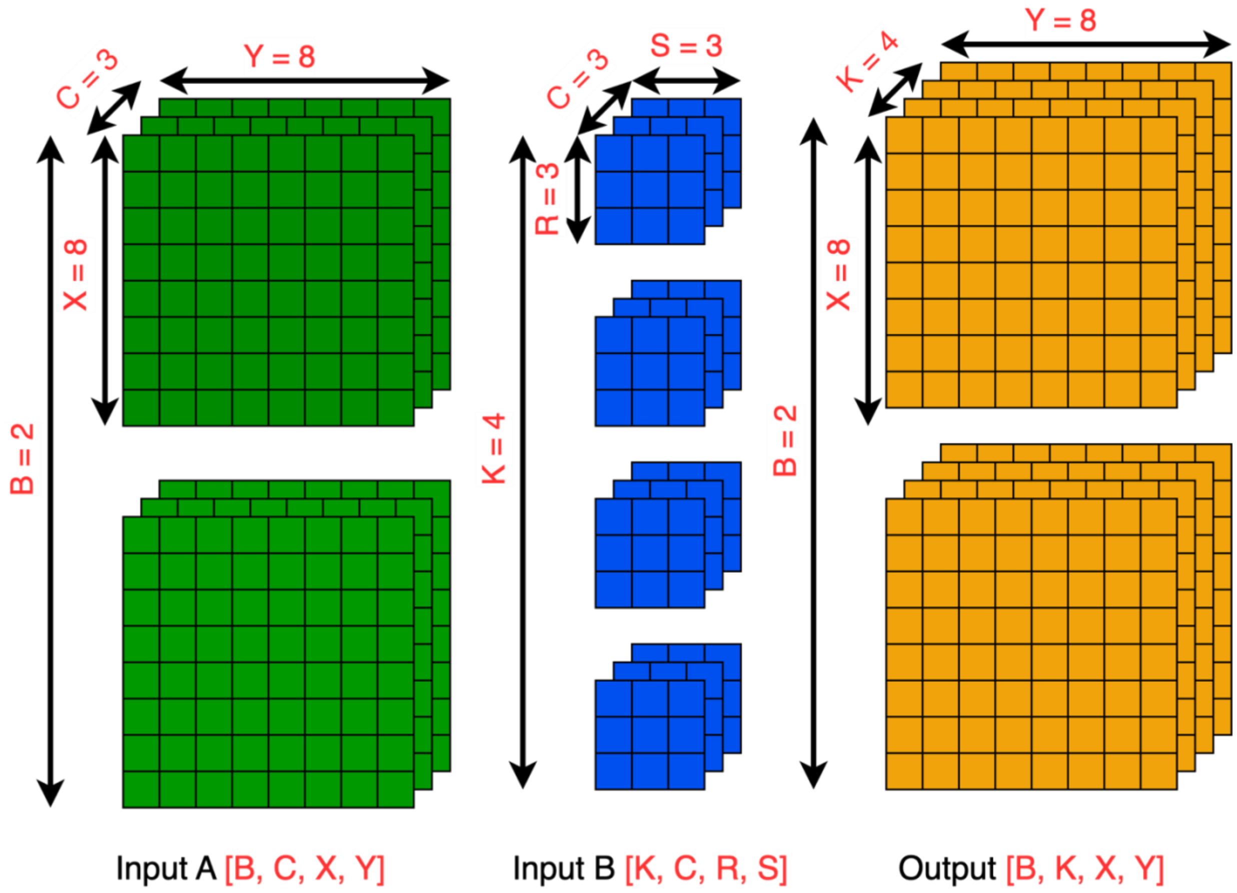


Addition and Multiplication is consider 1 operation each.

Please round the number of operations to the nearest full value.
Ex: Matrix of 1x100 and 100x1 will require 100 multiplications and 99 additions ~ round this to 100 + 100 = 200

A.1.iv) 2D Convolution

Conv2D(A, B) = C; Dim (A) = [B, C, X, Y], Dim (B) = [K, C, R, S], Dim (C) = [B, K, X, Y].
Assume enough padding so that output is of same size as the input.



After completing the code in operators.py, run cells in lab1A.ipynb till TODO A1.

Part A.2: Runtime Computations

For this part, you would be modifying the code in operator_base.py.

Before starting to write the function, please go to systems.py and get yourself familiarized with a system class, and various parameters associated to it.

```
A100_GPU = System( offchip_mem_bw=1935 GB/s,
                flops=312 TFLOPS, frequency=1095 MHz,
                compute_efficiency=0.75, memory_efficiency=0.7)
```

Shown above is an example declaration of system.

We will use these system parameters and operator values (num. of elements, num. of operators) to calculate the runtime of operators. Our final aim is to calculate the ideal roofline time of each operator for a given system.

We will do this in the following order:

A.2.i) Compute time

Use number of ops for given operator and various system parameters to determine the compute time.
System parameter op_per_sec shows how many operations can be calculated per second.
Note: while there are different type of operations (i.e. relu, addition, multiplication, etc.), for purpose of this lab.

A.2.ii) Memory time

Use number of elements for a given operator and various system parameters to determine the memory time. Assume datatype as BF16.

A.2.iii) Roofline time

Using the compute time and memory time, find the total execution time of each operator.

Assume the computation and memory operation is perfectly synchronized so they can be executed in parallel.

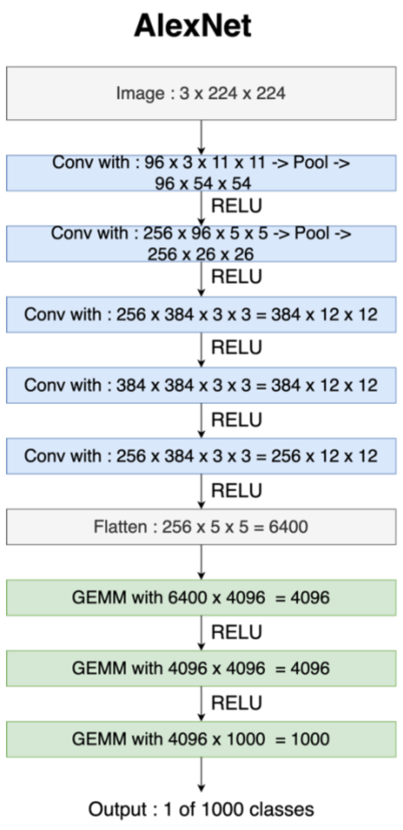
After completing the code in operator_base.py, run cells in lab1B.ipynb till TODO A1.

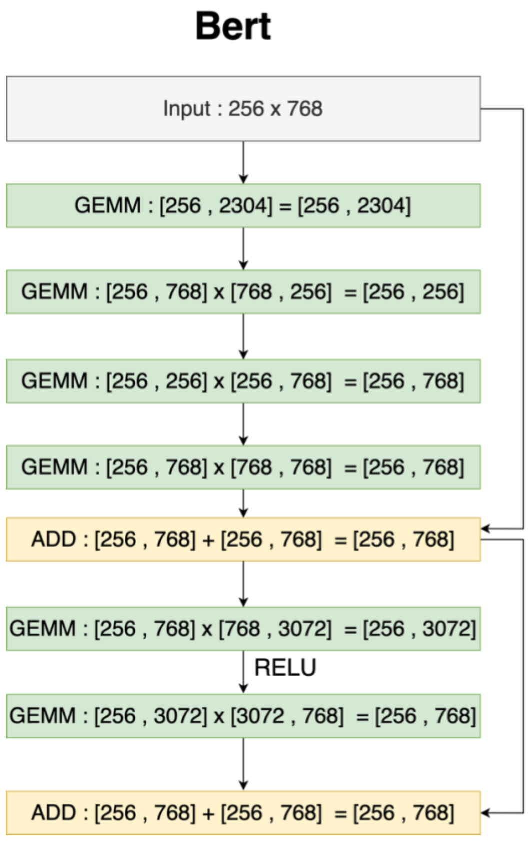
Part A.3: Building Neural Networks

Now that we have completed the operator definition, we are going build 2 networks, Alexnet and bert using the operators we defined.
We have defined the networks as series of conv, relu, gemm, and add. For this exercise, you can ignore the pool, and flatten operation.

Students need to fill the alexnet and Bert function in lab1a.ipynb. Please do use batch size as input for each model.

An example model declaration has also been shown in the notebook. Please use the same format to declare these 2 models.


AlexNet
Image : 3 x 224 x 224
Conv with : 96 x 3 x 11 x 11 -> Pool -> 96 x 54 x 54
RELU
Conv with : 256 x 96 x 5 x 5 -> Pool -> 256 x 26 x 26
RELU
Conv with : 256 x 384 x 3 x 3 = 384 x 12 x 12
RELU
Conv with : 384 x 384 x 3 x 3 = 384 x 12 x 12
RELU
Conv with : 256 x 384 x 3 x 3 = 256 x 12 x 12
RELU
Flatten : 256 x 5 x 5 = 6400
GEMM with 6400 x 4096 = 4096
RELU
GEMM with 4096 x 4096 = 4096
RELU
GEMM with 4096 x 1000 = 1000
Output : 1 of 1000 classes


Bert
Input : 256 x 768
GEMM : [256 , 2304] = [256 , 2304]
GEMM : [256 , 768] x [768 , 256] = [256 , 256]
GEMM : [256 , 256] x [256 , 768] = [256 , 768]
GEMM : [256 , 768] x [768 , 768] = [256 , 768]
ADD : [256 , 768] + [256 , 768] = [256 , 768]
GEMM : [256 , 768] x [768 , 3072] = [256 , 3072]
RELU
GEMM : [256 , 3072] x [3072 , 768] = [256 , 768]
ADD : [256 , 768] + [256 , 768] = [256 , 768]

A4. Comparing the performance of NN on different HWs

A4.i) Generate csv for alexnet and bert on jetson nano system, with batch size 4. Make sure to name the csv file *'output_a4i.csv'* and *'output_a4ii.csv'*

A4.ii) Observe whether the operators are memory bounded or compute bounded. Comment on the change in operator behavior between systems? Do they change, if so, why?

A4.iii) For running alexnet, what changes would you suggest to on hardware specs that would help in optimizing the performance?

Lab Submission

- Submission date: Feb 2nd
- Submission format: 1 zip + 5 csv.
    - output_a1.csv, output_a2.csv, output_a3i.csv, output_a3ii.csv, output_a4i.csv, output_a4ii.csv
    - Keep all csv in zip also