Georgia Institute of Technology

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

Spring 2025

Lab 1A

Due: Sunday, February 2, 2025 @ 11:59 pm EST

Contact for any queries: Ritik Raj (ritik.raj@gatech.edu)

Adopted from previous version created by Abhimanyu Bambhaniya

Instructions

Please read the following instructions carefully.

- Lab 1 is distributed in 2 parts (Each worth 8 points). This pdf only contains lab 1A.
- 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_GTID_ECE_8803_HML_sp25_lab1A.zip
  Note: Any other format will result in 1 point deduction (12.5% of the lab weightage)
- 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 SoftMax, Batch Normalization, Q/K/V Multiplication, Attention
- Calculate number of operations for SoftMax, Batch Normalization, Q/K/V Multiplication, Attention

Part A.2: Runtime Computations – 1 point

- Compute time
- Memory time
- Roofline time

Part A.3: Building Neural Networks - 1 point

- Llama
- gpt3

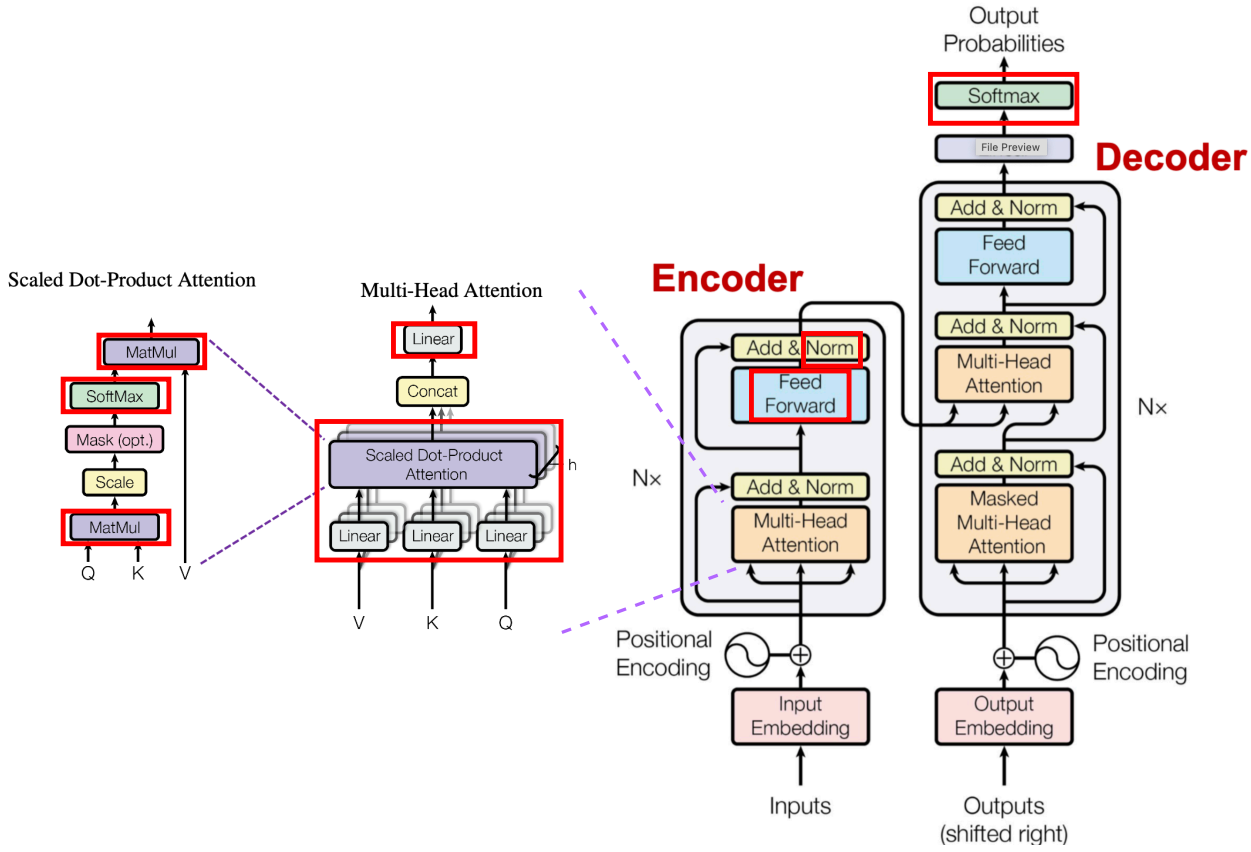
Part A.4: Comparing the performance of NN on different HWs - 2 points.

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



Encoder-Decoder Transformer (we will study operators bounded by red rectangle)

*A. Vaswani et al., ”Attention is all you need." NeurIPS, 2017.

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.

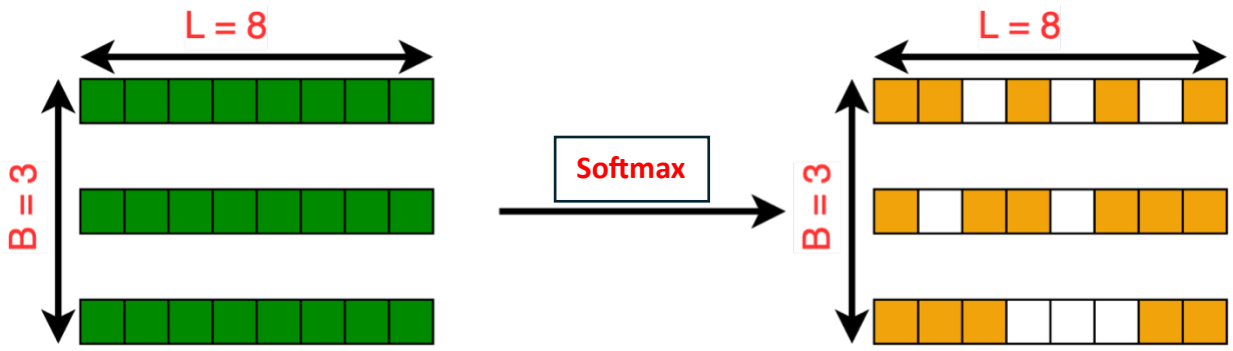
In get_tensors(), calculate the number of elements of input and elements of output for each operator.

In get_num_ops(), calculate the number of operations in each operator. Please don't count the number of operations multiple times to calculate the same thing.

Details of each of the 4 operator is explained in detail below. Assume addition, subtraction and multiplication count as one operation each while exponential function (e^x), division and square root count as (X+3) operations each where X is the last digit of your GT ID. For example, if your GTID is 903933092, then no. of operations for exponent, division and square root will count as 2+3=5.

A.1.i) SoftMax

For any vector x containing elements n elements x_0, x_1, ..., x_{n-1}
SoftMax of on element x_i is defined as $\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{n-1} e^{x_j}}$

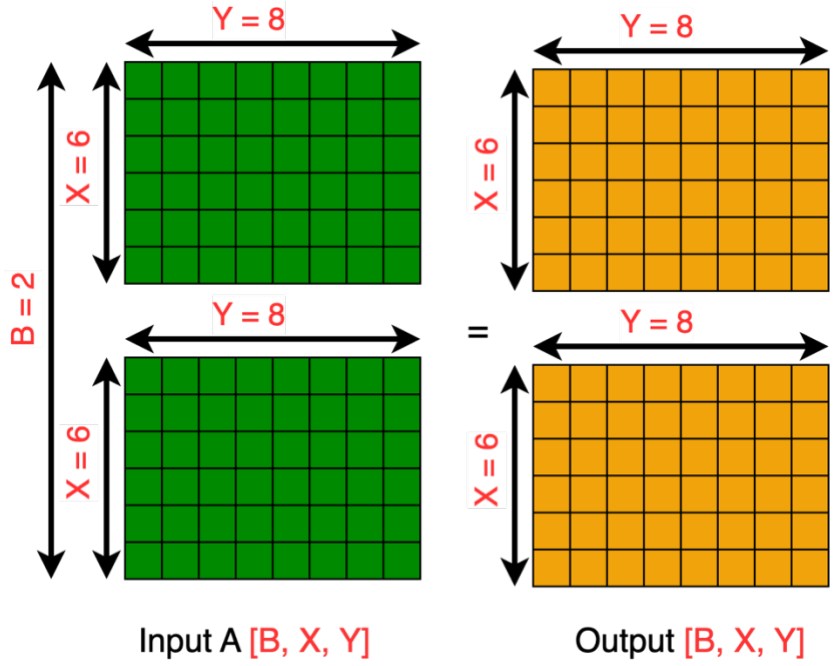


Note: Ignore the white boxes in the output, only focus on dimensions. They denote sparsity which arises out of high negative exponents which is insignificant for the purpose of this lab.

A.1.ii) Layer Normalization

Input A [B, X, Y] and Output [B, X, Y]
Layer Normalization is done across Y dimension i.e. each X vector in each batch is normalized separately. Assume each X vector contains Y no. of elements. Normalization of any element x in vector X is defined as:

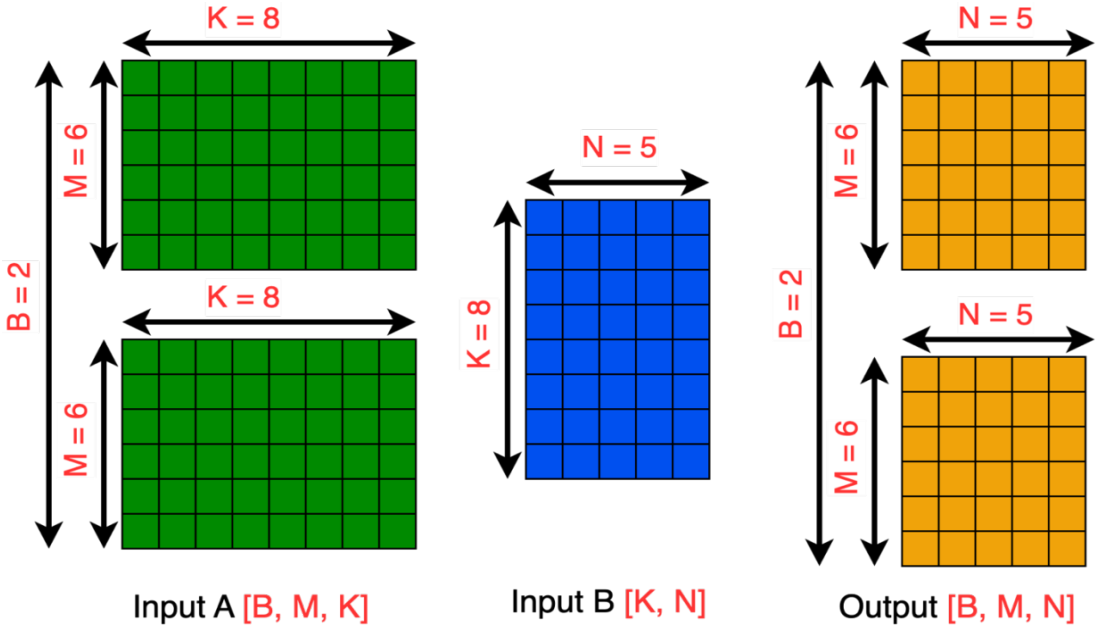
$$Norm(x) = \frac{x-\mu}{\sigma}, \mu\ (mean) = \frac{\sum_{j=0}^{Y-1} x_j}{Y}, \sigma\ (std) = \sqrt{\frac{\sum_{j=0}^{Y-1}(x_j-\mu)^2}{Y}}$$



Note: You need to count no. of operations to calculate mean and standard deviation as well.

A.1.iii) Matrix Multiplication

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



A.1.iv) Scaled-dot product Attention

Attention can be expressed in terms of SoftMax and GEMM operations.

GEMM1: $Q = X*W_Q$

Input X [B, M, K], Input W_Q [K, N], Output [B, M, N]

GEMM2: K = X*W_K
Input X [B, M, K], Input W_K [K, N], Output [B, M, N]

GEMM3: V = X*W_V
Input X [B, M, K], Input W_V [K, N], Output [B, M, N]

Attention (X) = SoftMax($\frac{Q*K^T}{\sqrt{c}}$) *V
Where c is a constant and $(.)^T$ represents matrix transpose

After completing the code in operators.py, run cells in lab1A.ipynb till 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)
```

Compute efficiency is a system parameter which lies in range (0,1].
The ideal system would have compute efficiency = 1.
Ex: If system is assigned 100 operations and has PEs to do 10 operations per cycle.
Number of cycles for system with compute efficiency of 1 = 100/10 = 10 cycles.
Number of cycle for system with compute efficiency of 0.5 = 100/10/0.5 = 20 cycles.
Number of cycle for system with compute efficiency of 0.25 = 100/10/0.25 = 40 cycles.
The same goes for memory efficiency as well.

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. The unit of compute time is seconds.

System parameter op_per_sec shows how many operations can be calculated per second.

A.2.ii) Memory time

Use number of elements (inputs and outputs) for a given operator and various system parameters to determine the memory time. Assume datatype as FP32.

Note: Ignore the no. of reads/writes for mean and standard deviation for layer norm and constant c for attention operation. Also, ignore the multiple reads/writes happening from local memory. Assume local memory capacity is large enough to ensure no capacity/conflict misses. Example: In GEMM, each input element is read multiple times from local memory but only read once from the main memory (DRAM/HBM). Only count reads/writes from the main memory.

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.

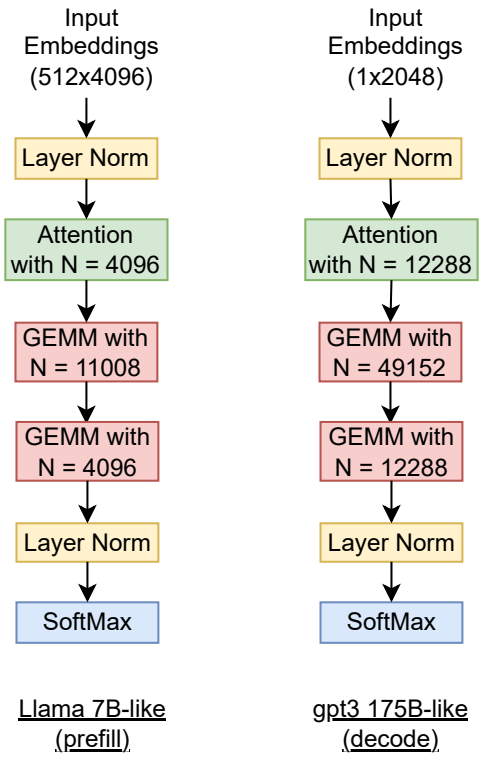
After completing the code in operator_base.py, run cells in lab1A.ipynb till A2.

Part A.3: Building Neural Networks

Now that we have completed the operator definition, we are going build 2 networks, llama and gpt3 using the operators we defined.

We have defined the networks as series of softmax, layer norm, gemm and attention.
Students need to fill the llama_7B_prefill and gpt3_175B_decode 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.



Please check part A.1 instructions to see what the N dimension mean for both GEMM and attention operators.

Run cells in lab1A.ipynb till A3.

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

A4.i) Generate csv for llama_7B_prefill and gpt3_175B_decode 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 gpt3_175B_decode, 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 + 6 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