Computer Architecture
2024 Spring

Final Project Part 2

Overview

Tutorial

- Gem5 Introduction
- Environment Setup

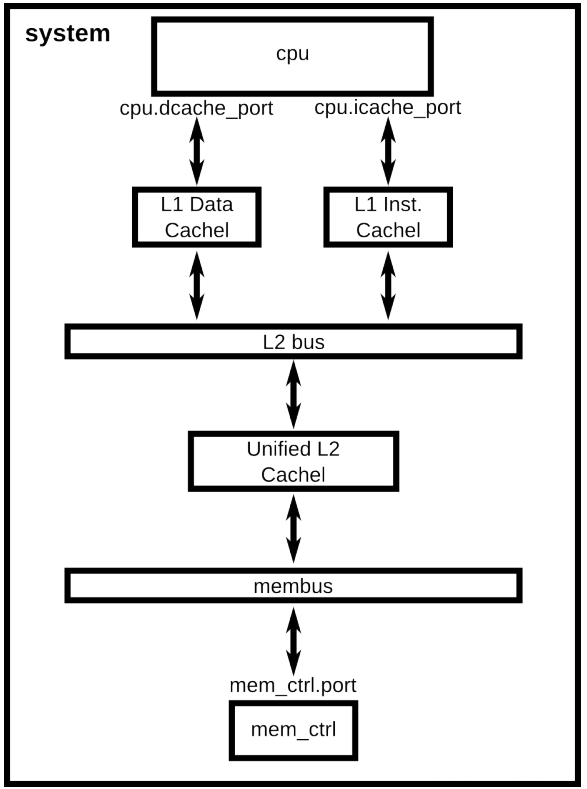
Projects

- Part 1 (5%)
  - Write C++ program to analyze the specification of L1 data cache.
- Part 2 (5%)
  - Given the hardware specifications, try to get the best performance for more complicated program.

Project 2

Description

In this project, we will use a two-level cache computer system. Your task is to write a ViT(**Vision Transformer)** in C++ and optimize it. You can see more details of the system specification on the next page.



System Specifications

- ISA: X86
- CPU: TimingSimpleCPU (no pipeline, CPU stalls on every memory request)
- Caches

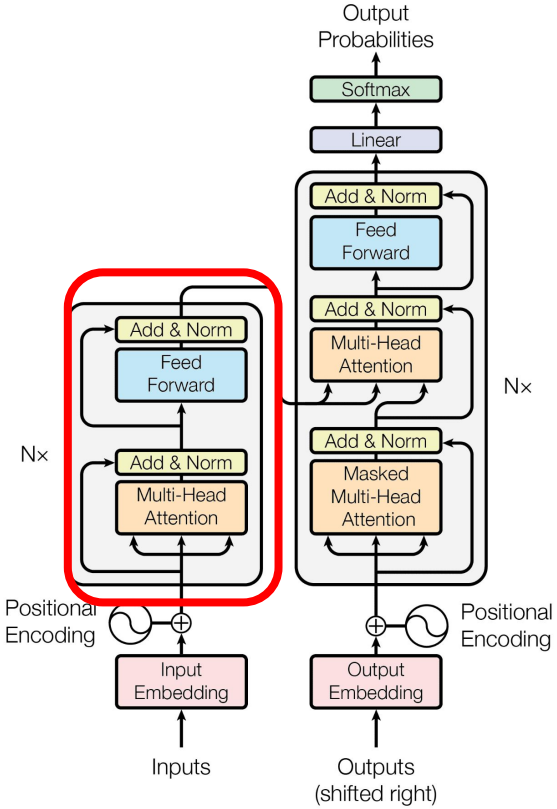
	I cache size	I cache associativity	D cache size	D cache associativity	Policy	Block size
L1 cache	16KB	8	16KB	4	LRU	32B
L2 cache	–	–	1MB	16	LRU	32B

* L1 I cache and L1 D cache connect to the same L2 cache

- Memory size: 8192MB

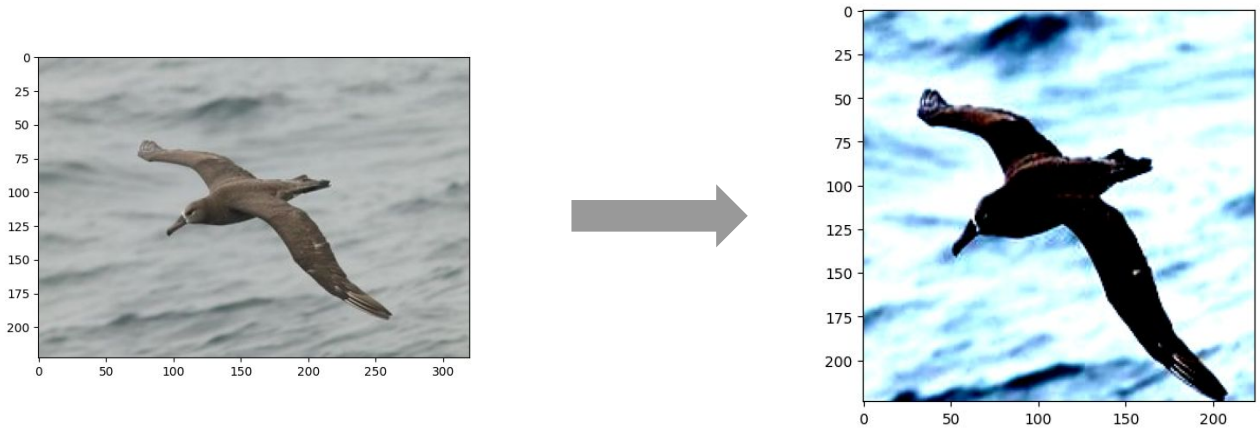
ViT(Vision Transformer) – Transformer Overview

- A basic transformer block consists of
  - Layer Normalization
  - MultiHead Self-Attention (MHSA)
  - Feed Forward Network (FFN)
  - Residual connection (Add)
- You only need to focus on how to implement the function in the **red box**
- **If you only want to complete the project instead of understanding the full algorithm about ViT, you can skip the section masked as red**



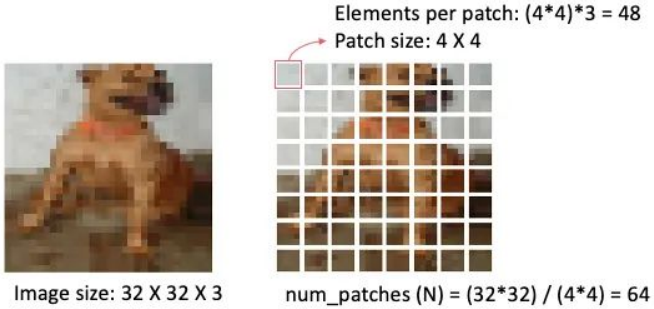
ViT(Vision Transformer) – Image Pre-processing

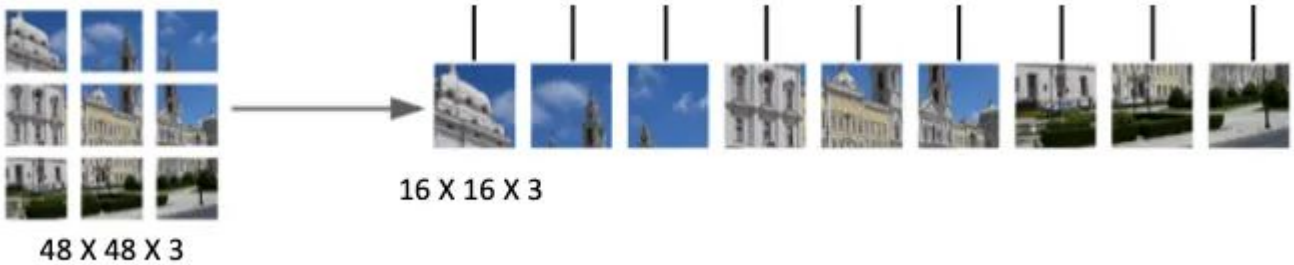
- Normalize, resize to (300,300,3) and center crop to (224,224,3)



ViT(Vision Transformer) – Patch Encoder

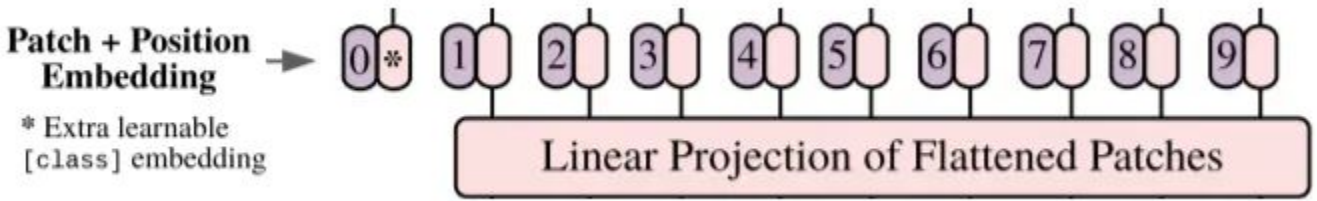
- In this project, we use Conv2D as Patch Encoder with kernel_size = (16,16), stride = (16,16) and output_channel = 768
- (224,224,3) -> (14,14, 16*16*3) -> (196, 768)





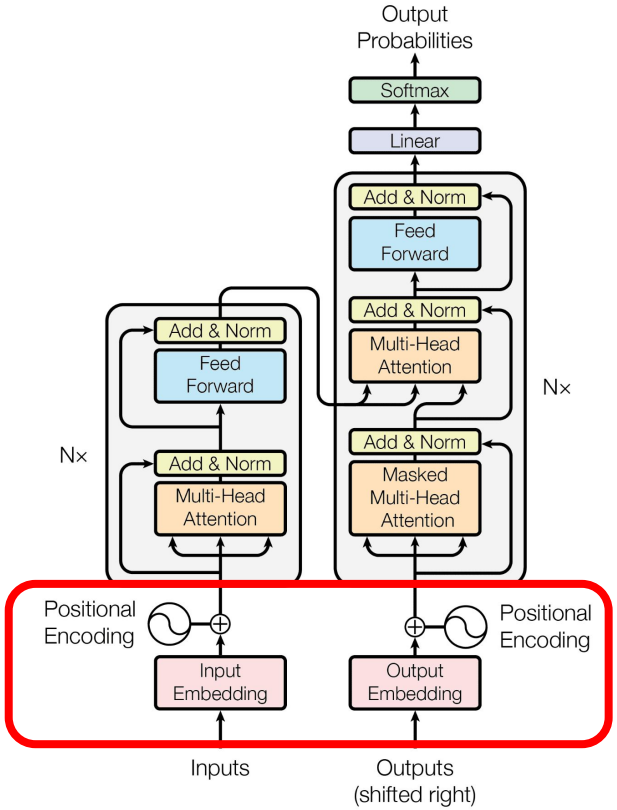
ViT(Vision Transformer) – Class Token

- Now we have 196 tokens and each token has 768 features
- In order to record global information, we need concatenate one learnable class token with 196 tokens
- (196,768) -> (197,768)



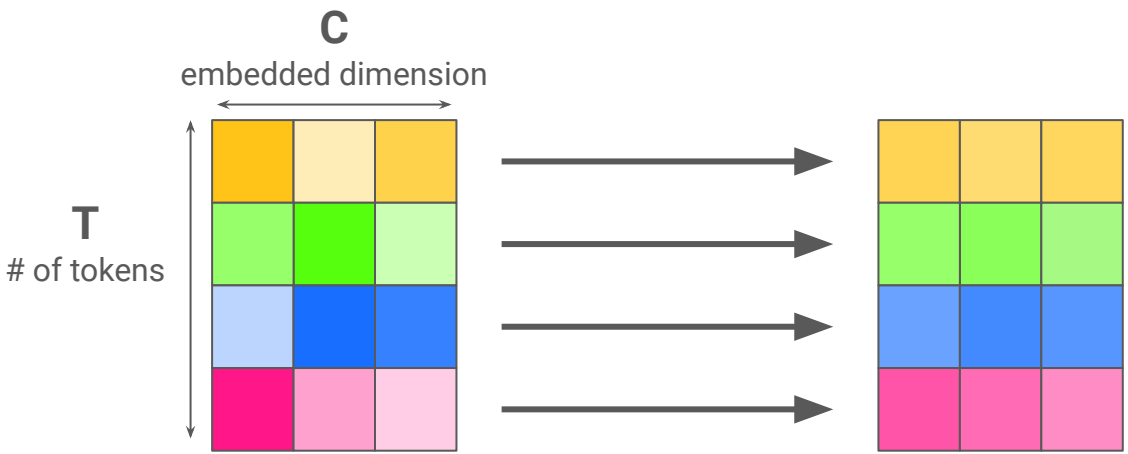
ViT(Vision Transformer) – Position Embedding

- Add the learnable position information on the patch embedding
- (197,768) + position_embedding(197,768) -> (197,768)



ViT(Vision Transformer) – Layer Normalization

- Normalize each token
- You need to normalize with the formula



$$\mu = \frac{1}{E}\sum_{i=1}^{E} x_i$$

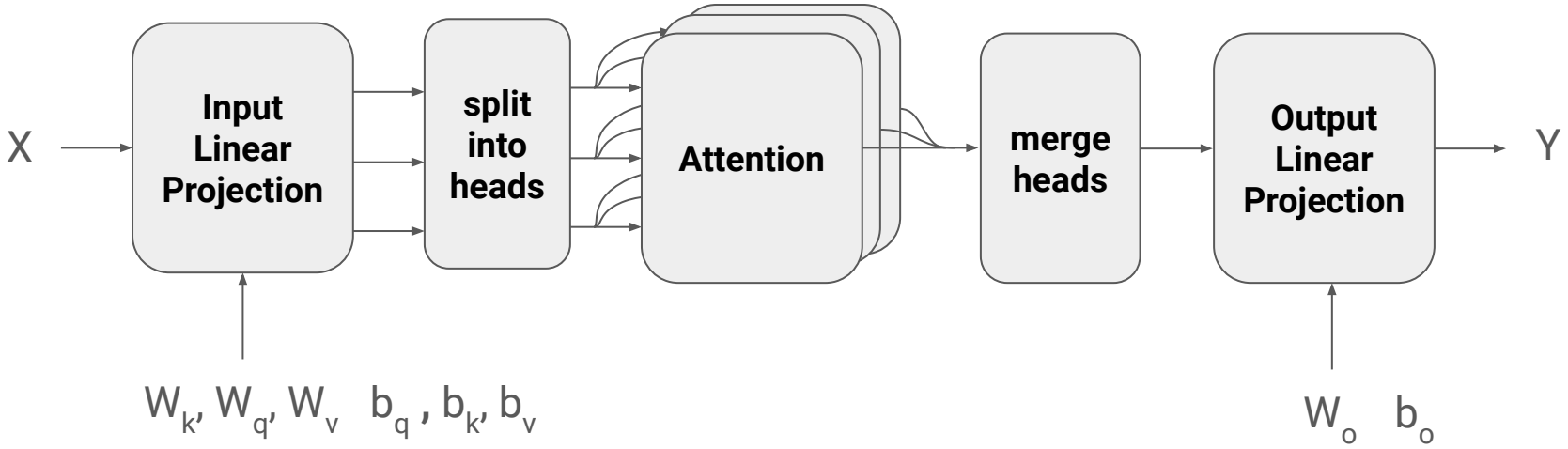
$$\sigma^2 = \frac{1}{E}\sum_{i=1}^{E}(x_i - \mu)^2$$

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$$y_i = \gamma\hat{x}_i + \beta$$

ViT(Vision Transformer) – MultiHead Self Attention (1)

- $W_k, W_q, W_v \in R^{c \times c}$
- $b_q, b_k, b_v \in R^c$
- $W_o \in R^{c \times c}$
- $b_o \in R^c$



ViT(Vision Transformer) – MultiHead Self Attention (2)

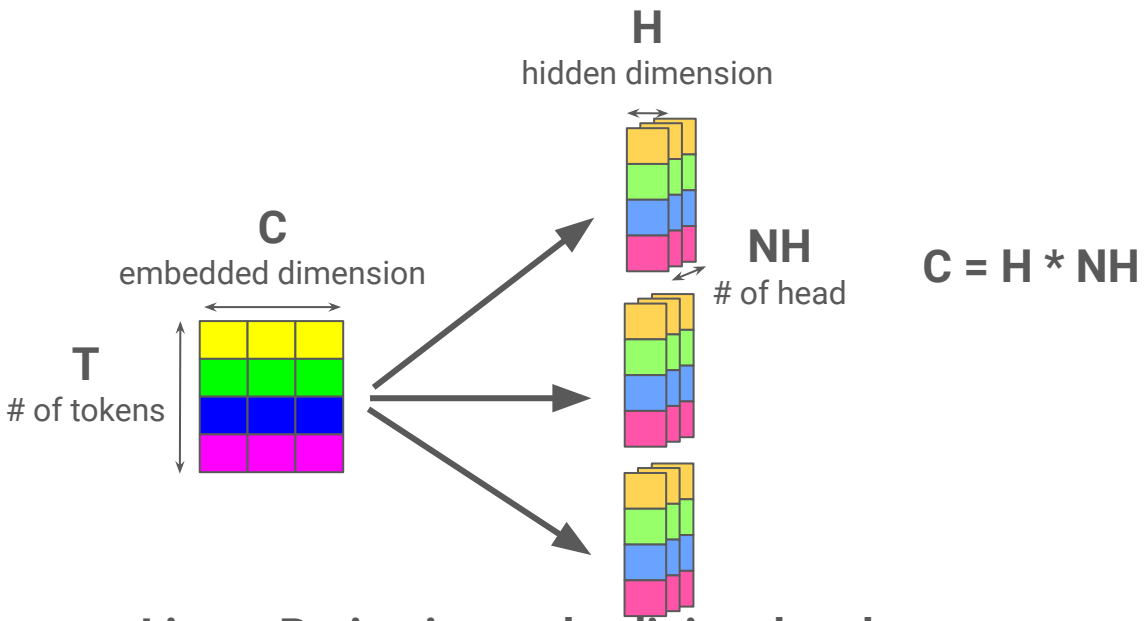
- Get Q, K, V $\in R^{T \times (NH*H)}$ after input linear projection
- Split Q, K, V into $Q_1, Q_2, Q_3, \ldots, Q_{NH}$ $K_1, K_2, K_3, \ldots, K_{NH}$ $V_1, V_2, V_3, \ldots, V_{NH} \in R^{T \times H}$

Linear Projection

$$Q = XW_q^T + b_q$$

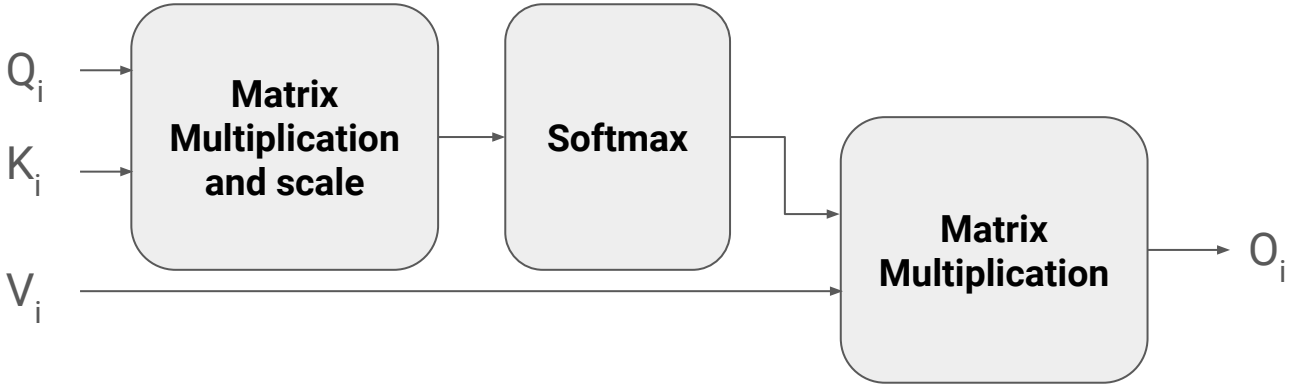
$$K = XW_k^T + b_k$$

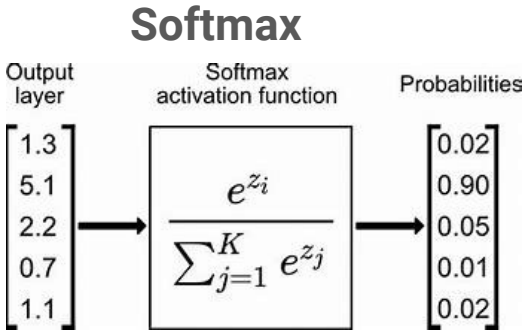
$$V = XW_v^T + b_v$$



ViT(Vision Transformer) – MultiHead Self Attention (2)

- For each head i, compute $S_i = Q_i K_i^T / \text{square\_root}(H) \in R^{T \times T}$
- $P_i = \text{Softmax}(S_i) \in R^{T \times T}$, **Softmax is a row-wise function**
- $O_i = P_i V_i \in R^{T \times H}$



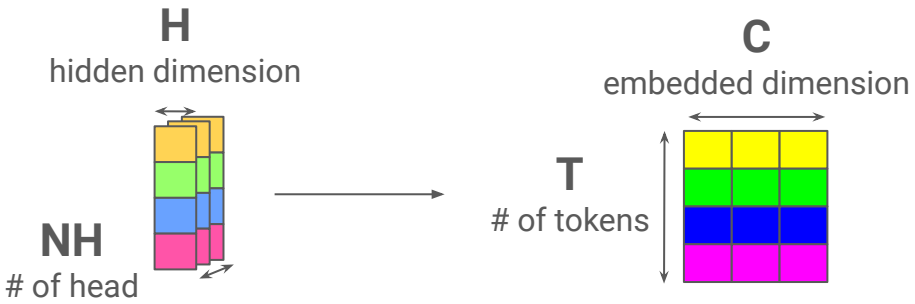


ViT(Vision Transformer) – MultiHead Self Attention (3)

- $O_i \in R^{T \times H}$, $O = [O_1, O_2, \ldots, O_2]$

Linear Projection

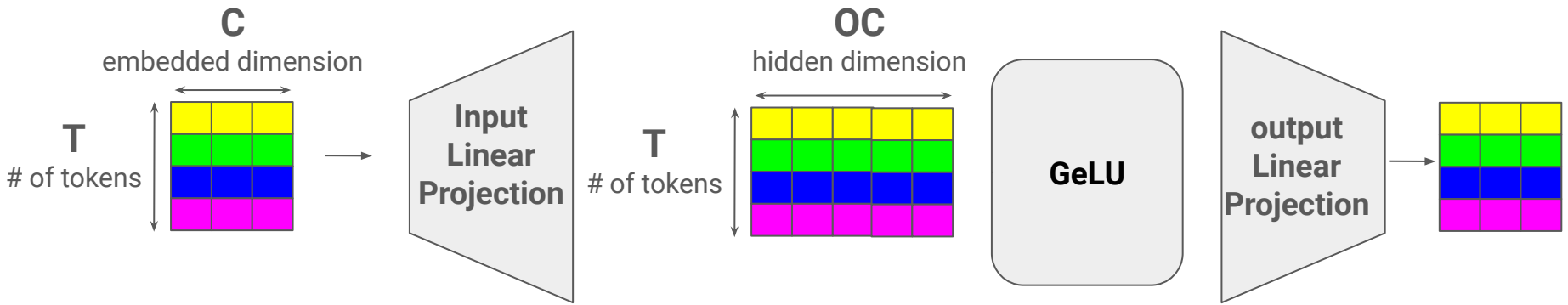
$$output = OW_o^T + b_o$$



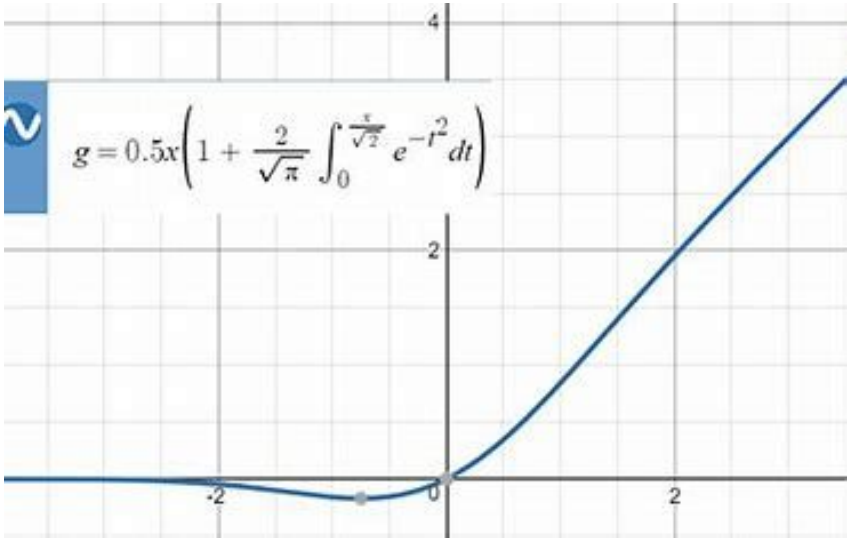
merge heads and Linear Projection

ViT(Vision Transformer) – Feed Forward Network

- Get Q, K, V $\in R^{T \times (h*H)}$ after input linear projection
- Split Q, K, V into $Q_1, Q_2, Q_3, \ldots, Q_h$ $K_1, K_2, K_3, \ldots, K_h$ $V_1, V_2, V_3, \ldots, V_h \in R^{T \times H}$



ViT(Vision Transformer) – GeLU



- $GELU(x) = 0.5x(tanh[\sqrt{2/\pi}\,(x + 0.044715x^3)])$
- $GELU(x) = x\sigma(1.702x)$

ViT(Vision Transformer) – Classifier

- Contains a Linear layer to transform 768 features to 200 class
  - (197, 768) -> (197, 200)
- Only refer to the first token (class token)
  - (197, 200) -> (1, 200)

ViT(Vision Transformer) – Work Flow

gem5

$ make MHSA_tb

layernorm

$ make layernorm_tb

matmul

$ make matmul_tb

gelu

$ make gelu_tb

$ make feedforward_tb

matmul
attention
matmul

matmul
gelu
matmul

layernorm
MHSA
+
layernorm
FFN
+

$ make transformer_tb

Pre-pocessing

Embedder

Transformer x12

layernorm

Classifier

Argmax

Black footed Albatross

Load_weight

m5_dump_init

layernorm
MHSA
residual

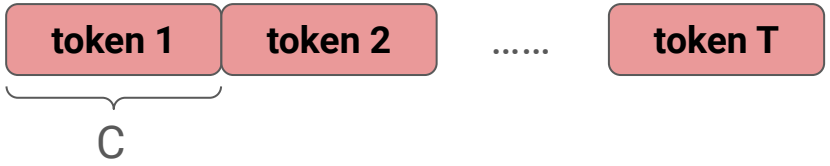
m5_dump_stat

$ run_all.sh

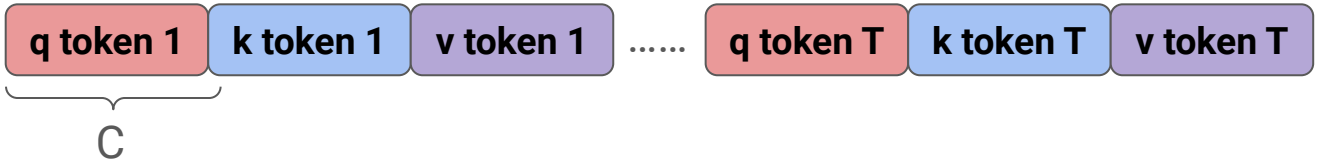
ViT(Vision Transformer) – Shape of array

layernorm input/output [T*C]

MHSA input/output/o [T*C]

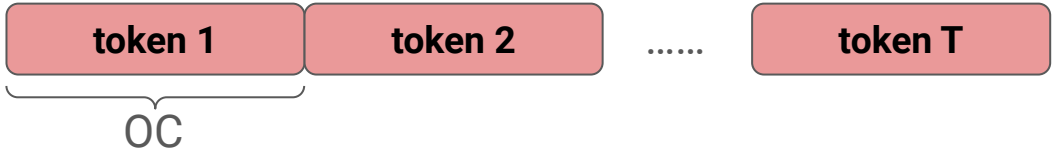


MHSA qkv [T*3*C]



feedforward input/output [T*C]

feedforward gelu [T*OC]



Common problem

- Segmentation fault
  - ensure that you are not accessing a nonexistent memory address
  - Enter the command $ulimit -s unlimited

All you have to do is

- Download TA’s Gem5 image
  - docker pull yenzu/ca_final_part2:2024
- Write C++ with understanding the algorithm in ./layer folder
  - make clean
  - make <layer>_tb
  - ./<layer>_tb

All you have to do is

- Ensure the ViT will successfully classify the bird
  - python3 embedder.py --image_path images/Black_Footed_Albatross_0001_796111.jpg --embedder_path weights/embedder.pth --output_path embedded_image.bin
  - g++ -static main.cpp layer/*.cpp -o process
  - ./process
  - python3 run_model.py --input_path result.bin --output_path torch_pred.bin --model_path weights/model.pth
  - python3 classifier.py --prediction_path torch_pred.bin --classifier_path weights/classifier.pth
  - After running the above commands, you will get the following top5 prediction.

```
classified result:  tensor([ 0,  2,  1, 71,  7])
```

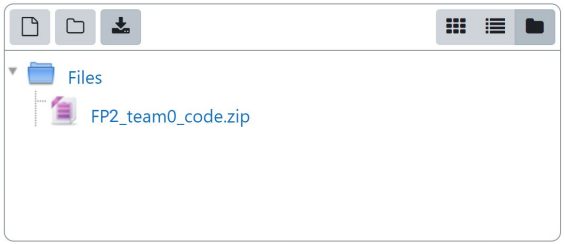
- Evaluate the performance of part of ViT, that is layernorm+MHSA+residual
  - Need about 3.5 hours to finish the simulation
  - Check stat.txt

Grading Policy

- (50%) Verification
  - (10%) matmul_tb
  - (10%) layernorm_tb
  - (10%) gelu_tb
  - (10%) MHSA_tb
  - (10%) transformer_tb
- (50%) Performance
  - max(sigmoid((27.74 - student latency)/student latency))*70, 50)
- You will get 0 performance point if your design is not verified.

Submission

- Please submit code on E3 before 23:59 on **June 20, 2024**.
- Format
  - Code: please put your code in a folder named **FP2_team<ID>_code** and compress it into a zip file.



- **Late submission is not allowed.**
- **Plagiarism is forbidden, otherwise you will get 0 point!!!**

FP2_team<ID>_code folder

- You should attach the following documents
  - matmul.cpp
  - layernorm.cpp
  - gelu.cpp
  - attention.cpp
  - residual.cpp