The project outline

1. **The background and requirement**

Transformer has emerged as a popular deep neural network (DNN) model for Neural Language Processing (NLP) applications and demonstrated excellent performance in neural machine translation, entity recognition, etc. However, its scaled dot-product attention mechanism in auto-regressive decoder brings a performance bottleneck during inference. Transformer is also computationally and memory intensive and demands for a hardware acceleration solution. Although researchers have successfully applied ReRAM based Processing-in-Memory (PIM) to accelerate convolutional neural networks (CNNs) and recurrent neural networks (RNNs), the unique computation process of the scaled dot-product attention in Transformer makes it difficult to directly apply these designs. Besides, how to handle intermediate results in Matrix-matrix Multiplication (MatMul) and how to design a pipeline at a finer granularity of Transformer remain unsolved. In this work, we propose ReTransformer – a ReRAM-based PIM architecture for Transformer acceleration. ReTransformer can not only accelerate the scaled dot-product attention of Transformer using ReRAM-based PIM but also eliminate some data dependency by avoiding writing the intermediate results using the proposed matrix decomposition technique.

1. **The work introduction**

Build one transformer model use Pytorch, then trying to recode it instead of Pytorch but the pure Python, and especially for the multiplication part “scaled dot-product”, I will extract this part and pay more attention on the calculation power consumption more easily to understand is the waste of computing time, More details I need to get into each encode layer and decode layer both of them has 6 layers and obtain the computing time, power consumption of every sublayers. Then we will do this accelerate by the hardware ReRAM based to eliminate some data dependency by avoiding writing the intermediate results using the proposed matrix decomposition technique

**3.The technical details.**

* Principle
* Setting environment.
* Step details

Principle

By the use of heterogeneity, the hardware-software cooperation architecture can take advantages of both hardware and software approach and achieve both generalization and hardware-acceleration. For RRAM based processing for Transformer acceleration.

Setting environment

Here I used software environment python 3.8

Hardware environment GPU 3080.

We will use pytorch to accomplish the whole project.

The import list:

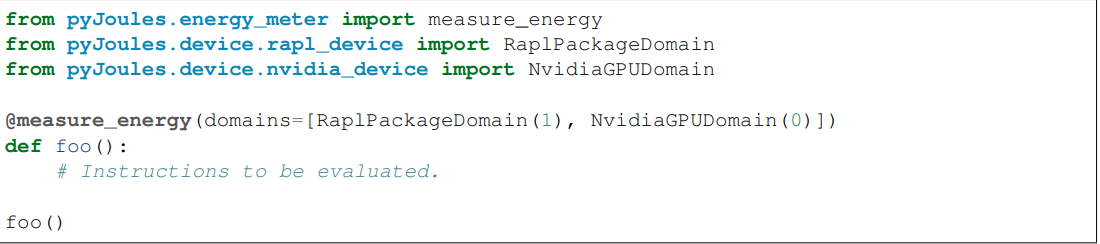
import torch  
import torch.nn as nn  
import math  
from torch.autograd import Variable  
import torch.nn.functional as F  
import copy  
from pyJoules.energy\_meter import measure\_energy  
from pyJoules.device.nvidia\_device import NvidiaGPUDomain  
from pyJoules.handler.csv\_handler import CSVHandler

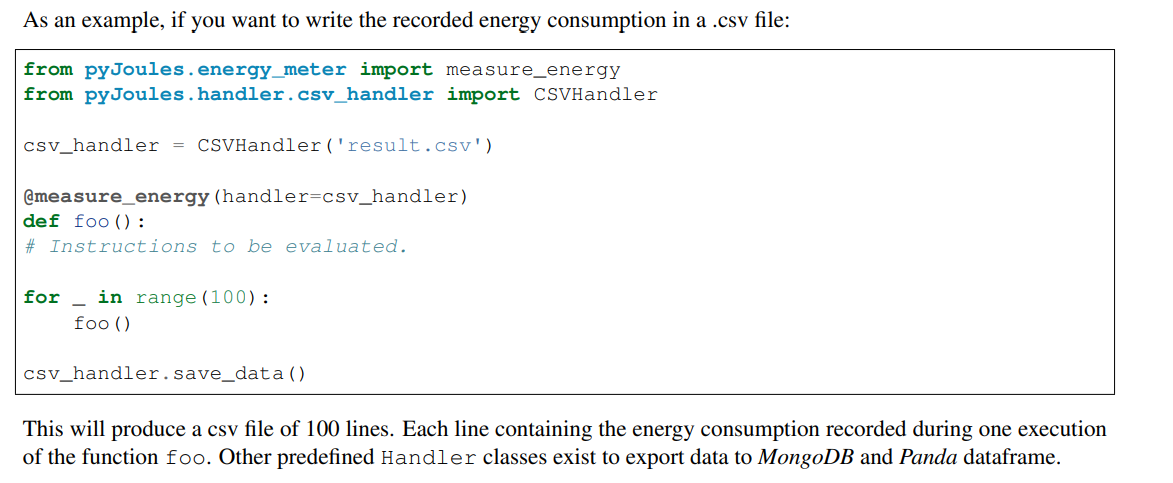
especially here the pyJoules, this package will give support us the method to get function computing time and power consumption. And also can save the experimental data into one csv file.

And pyJoules python3.7 version or above is required.

You can use pip install to accomplish these packages.

THE USAGE EXAMPLES BELOW FOR pyJoules





Step details

The reconstructed transformer model, will use several new functions I wrote.

there are whole files of project.

1.main.py

2.each\_layer\_time.py

3.soft\_matmul.py

4.ave\_time\_pow.py

5.multiply.py

# ## I will explain each functions below with index number.

I will introduce you guys one good package 'pyJoules', this package will exactly give you the function computing time and powe consumption. And also can save the experimental data into one csv file. (Just google it, you can find more information)

[1] We mannully initialized x for embedding and positional embedding.

and take the output as encoder input. Then do 1000 times running to calculate the time and power consumption. same as the decoder part. At the end it will generate 5 .csv files which will contain the whole data for each sublayers.

[2] This is the transfomer model, there are some new fucnitons, softmax, Lin, matmul\_for\_looop, the\_multiply\_for . Also we will replace the every torch.Linear and torch.softmax with softmax and Lin. we will do the torch.matmul by the designed two for loop functions.

[3] Used to test the softmax and matmul\_for\_loop functions independently for thousand times. and get the computing time and power consumption then save it in their own csv file.

[4]This one will manage the cvs file. collecting the data and calculate the average computing time and the average power consumption. You will see the comments. And in data\_multi\_pc.csv , data\_layer\_norm.csv, data\_feed\_forward.csv, they has encoder part data, and decoder part data. See more details in my code comments.

[5] The redesigned dot product method. There are two function inside,

one ths the\_multiply\_for, used to calculate the dot product.

other one is matmul\_for\_loop, used to calculate matrix multiplication.