ReTransformer

ReRAM-based Processing-in-Memory Architecture for Transformer Acceleration

Reza Adinepour

Amirkabir University of Technology (Tehran Polytechnic)

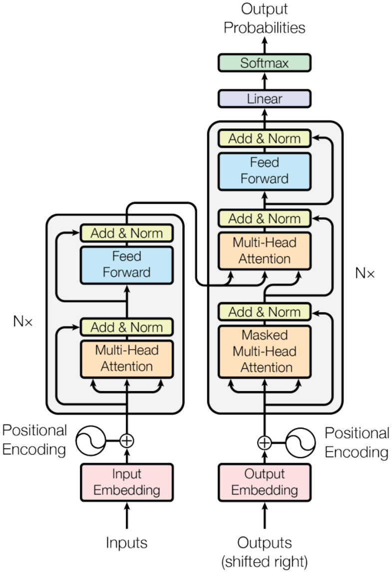
adinepour@aut.ac.ir

Computer Engineering Department
June 27, 2024

Presentation Overview

What is a Transformer Network?

1. Introduced in **"Attention is All You Need"** (2017)
2. Unlike traditional RNNs and LSTMs, Transformers **do not** process data sequentially but use a mechanism called **"self-attention"** to draw global dependencies between input and output.



Why Use Transformers?

1. **Parallelization:**
   Unlike RNNs, Transformers can process input data in parallel, leading to faster training times.
2. **Self-Attention Mechanism:**
   This allows the model to weigh the importance of different words in a sentence, capturing long-range dependencies more effectively.
3. **Scalability:**
   Transformers can be scaled up effectively, leading to improved performance with more data and larger models.
4. **Versatility:**
   They are used in various applications, from machine translation and text generation to image processing and more.

Strengths of Transformer

1. **Efficiency:**
   Due to parallel processing, they train faster on large datasets.
2. **Accuracy:**
   State-of-the-art performance in many tasks, particularly in NLP.
3. **Flexibility:**
   Applicable to a wide range of tasks beyond language, such as image and speech processing.
4. **Transfer Learning:**
   Pre-trained models like BERT and GPT can be fine-tuned for specific tasks with relatively small amounts of data.

Weaknesses of Transformers

1. **Resource-Intensive:**
   Require significant computational power and memory, especially for large models.
2. **Complexity:**
   More challenging to understand and implement compared to simpler models.
3. **Data Requirements:**
   Performance often hinges on the availability of large-scale datasets for pre-training.
4. **Inference Speed:**
   Performance bottlenecks during inference due to the scaled dot-product attention mechanism.

Motivation

In this paper:

1. Developed **ReTransformer, a ReRAM-based Processing-in-Memory (PIM) architecture** specifically designed to accelerate Transformer models.
2. Implemented **optimized MatMul operations** to reduce data dependency and intermediate result handling
3. Designed a hybrid softmax mechanism combining in-memory logic and look-up tables for efficient softmax calculations.
4. Introduced a **sub-matrix pipeline design** for better utilization of ReRAM crossbars and improved throughput.

Motivation

And Improvements Made is:

1. **Computing Efficiency:**
   23.21x improvement over GPU, 3.25x over PipeLayer.
2. **Power Consumption:**
   1086x reduction compared to GPU, 2.82x compared to PipeLayer.
3. **Latency Reduction:**
   1.32x for smaller models, 1.16x for larger models.
4. **Softmax Efficiency:**
   32% lower power consumption compared to traditional CMOS-based designs.
5. **Throughput Enhancement:**
   1.18x increase in computational throughput.

Motivation

This concept use in CNNs and RNNs. but we can't and cant be directly applied to Transformer due to the following reasons:

1. **Matrix-Matrix Multiplication:**
   Transformers require frequent matrix-matrix multiplications, causing potential slowdowns and reduced efficiency due to intermediate result storage.
2. **Different Computations:**
   Unlike CNNs, Transformers use scaled dot-product attention, necessitating different computational approaches.
3. **Finer Pipeline Granularity:**
   Transformer accelerators need a more detailed pipeline design compared to the layer-level granularity used in previous designs.

Non-uniform Memory Access (NUMA) (Cont.)

Advantages:

1. **Higher Implementation Complexity:** Additional hardware and software complexity.
2. **Higher Latency for Remote Access:** Accessing remote memory incurs higher latency.

Example System:

1. **Multi-Socket Server:**
    1. Each socket has its processors and memory banks.
    2. Sockets connected via a high-speed interconnect.
    3. Commonly used in data centers, offers better performance for specific workloads.

Code Snippets

Run the code!
— This code is generated by ChatGP

Differences between UMA and NUMA

1. **Memory access time:**
   1. NUMA:
      Memory access time varies depending on the location of the data in memory. Accessing data in the local memory of a processor is faster than accessing data in the memory of a remote processor.
   2. UMA:
      Memory access time is uniform across all processors since they share the same memory pool.
2. **Scalability:**
   1. NUMA architecture is highly scalable and can support a large number of processors.
   2. UMA architecture is not as scalable as NUMA and may face performance issues when used with a large number of processors.

The End

Questions? Comments?

You can find this slides here:

github.com/M-Sc-AUT/M.Sc-Computer-Architecture/Memory Technologies