Next Generation AI: Transformers and Hyper-Dimensional Computing

Integrated Systems Laboratory (ETH Zürich)

Gamze İslamoğlu  gislamoglu@iis.ee.ethz.ch
Manuel Eggimann  meggimann@iis.ee.ethz.ch

PULP Platform
Open Source Hardware, the way it should be!
ChatGPT isn’t so great!

Who is the leader of the PULP group?

As of my knowledge cutoff in September 2021, Professor Luca Benini was one of the key leaders of the PULP (Parallel Ultra Low Power) project. However, please note that leadership roles can change over time. To obtain the most accurate and up-to-date information on the current leader of the PULP group, I recommend visiting the official PULP project website or reaching out to the project directly for the latest information.

- Up-to-date information
- Knowledge of your data
- More challenging reasoning
ChatGPT isn’t so great!

Revolutionize your Enterprise Data with ChatGPT: Next-gen Apps w/ Azure OpenAI and Cognitive Search

AWS Machine Learning Blog

Question answering using Retrieval Augmented Generation with foundation models in Amazon SageMaker JumpStart

By Xin Huang, Rachna Chada, Hemant Singh, Ashish Khetan, Manas Dadarkar, and Kyle Ulrich | on 02 MAY 2023 | in Amazon SageMaker, Amazon SageMaker JumpStart, Artificial Intelligence, Intermediate (200) | Permalink | Comments

Today, we announce the availability of sample notebooks that demonstrate question answering tasks using a Retrieval Augmented Generation (RAG)-based approach with large language models (LLMs) in Amazon SageMaker JumpStart. Text generation using RAG with LLMs enables you to generate domain-specific text outputs by supplying specific external data as part of the context fed to LLMs.

• Up-to-date information
• Knowledge of your data
• More challenging reasoning
Retrieval Augmented Generation (RAG)

Vector search

Databases

Representation of question (embeddings)

Question

Relevant information for enhanced context

Large Language Models
Retrieval Augmented Generation (RAG)

Ultra-efficient HW for next generation AI

Representation of question (embeddings)
Relevant information for enhanced context

Vector search

Question

Databases

Hyper-Dimensional Computing

Large Language Models

Transformers
Why Transformers?

- **Versatility**
  - Natural language processing, computer vision, robotics, biology, ...

- **Homogenization of models**
  - Transformers as *foundation models*!

- **Transfer learning**
  - Train on a large-scale dataset and fine-tune on specific tasks with smaller datasets.
Attention is all you need!

The animal didn't cross the street because it was too tired.
Attention but how?

Add & Norm

Feed Forward

Add & Norm

Multi-HeadAttention

Linear

MatMul

Softmax

MatMul

Query

Key

Value

Linear

Linear

Linear

I love PULP!
Challenges in **Attention**

- Attention matrix is a square matrix of order input length.
  - Computational complexity
  - Memory requirements
Challenges in *Attention*

- Attention matrix is a square matrix of order input length.
  - Computational complexity
  - Memory requirements

- Every attention layer applies *Softmax* to attention matrix!
Challenges in **Attention**

- Attention matrix is a square matrix of order input length.
  - Computational complexity
  - Memory requirements

- Every attention layer applies **Softmax** to attention matrix!
  - 3 passes over a row.
  - Quantization is problematic.

$$\text{Softmax}(x)_i = \frac{e^{x_i - \max(x)}}{\sum_{j=1}^{n} e^{x_j - \max(x)}}$$
ITA: Integer Transformer Accelerator

- **Attention** accelerator for transformers!
- INT8 quantized networks
- Output stationary - Local weight stationary
  - Spatial input reuse
  - Spatial output partial sum reuse
- Special **Softmax** unit!
ITA - Architecture

$N = 16$ dot product units that compute the dot product between two vectors of $M = 64$ elements

$N \times 2M = 2$ KiB
Hardware-friendly \textit{Softmax}

\[
\text{Softmax}(x)_i = \frac{e^{x_i - \max(x)}}{\sum_j^n e^{x_j - \max(x)}}
\]

\[
\text{Softmax}(x)_i = \frac{1}{\sum_j^n 2^{(x_{qi} - \max(x_q)) \gg 5}} 2^{(x_{qi} - \max(x_q)) \gg 5}
\]
Hardware-friendly \textit{Softmax}

Directly operates on quantized values.

No exponentiation modules and multipliers.

Computes softmax on streaming data.

\[
\text{Softmax}(x)_i = \frac{1}{\sum_j^n 2^{(x_{qi} - \max(x_q)) > 5}} 2^{(x_{qi} - \max(x_q)) > 5}
\]
Hardware-friendly *Softmax*

\[
\text{Softmax}(x)_i = \frac{1}{\sum_j 2^{(x_{qj} - \max(x_q)) \gg 5}} 2^{(x_{qi} - \max(x_q)) \gg 5}
\]
Hardware-friendly **Softmax**

\[
\text{Softmax}(x)_i = \frac{1}{\sum_j 2^{(x_{qj} - \max(x_q)) \gg 5}} 2^{(x_{qi} - \max(x_q)) \gg 5}
\]
Hardware-friendly **Softmax**

\[
\text{Softmax}(x)_i = \frac{1}{\sum_j^n 2^{(x_{qj} - \max(x_q))\gg 5}} \times 2^{(x_{qi} - \max(x_q))\gg 5}
\]
Hardware-friendly *Softmax*

\[
\text{Softmax}(x)_i = \frac{1}{\sum^n_{j} 2^{(x_{qj} - \max(x_q)) \gg 5}} 2^{(x_{qi} - \max(x_q)) \gg 5}
\]
Hardware-friendly \textit{Softmax}

\[
\text{Softmax}(\mathbf{x})_i = \frac{1}{\sum_j^n 2^{(x_{qj} - \max(x_q))} \gg 5} 2^{(x_{qi} - \max(x_q))} \gg 5
\]
Physical Implementation

• **Implemented in GF22**
  • Target frequency of 500 MHz (SS/0.72V/125°C)

• **Area of 0.17 mm²**
  • Softmax module has only **3.3%** area contribution, corresponding to 28.7 KGE.

• **Power of 60 mW** (TT/0.80V/25°C)
  • Softmax module consumes **1.4%** of the power.
# Comparison to State-of-the-Art

<table>
<thead>
<tr>
<th></th>
<th>Wang et al. (\textit{ISSCC’22})</th>
<th>Keller et al. (\textit{VLSI-TC’22})</th>
<th>ITA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology [nm]</td>
<td>28</td>
<td>5</td>
<td>22</td>
</tr>
<tr>
<td>Voltage [V]</td>
<td>0.56-1.1</td>
<td>0.46-1.05</td>
<td>0.8</td>
</tr>
<tr>
<td>Frequency [MHz]</td>
<td>50-510</td>
<td>152-1760</td>
<td>500</td>
</tr>
<tr>
<td>Data formats</td>
<td>INT8</td>
<td>INT8</td>
<td>INT8</td>
</tr>
<tr>
<td>Throughput [TOPS]</td>
<td>0.52-4.07*</td>
<td>1.8</td>
<td>1.02</td>
</tr>
<tr>
<td>Energy efficiency [TOPS/W]</td>
<td>1.91-27.56*</td>
<td>39.1</td>
<td>51.1</td>
</tr>
<tr>
<td>Area efficiency [TOPS/mm²]</td>
<td>0.076-0.597*</td>
<td>11.7</td>
<td>5.93</td>
</tr>
<tr>
<td>Area efficiency [TOPS/MGE]</td>
<td>0.025-0.192*</td>
<td>6.1x</td>
<td>1.18</td>
</tr>
</tbody>
</table>

*90% sparsity.
Comparison to a software baseline on MemPool

- **Performance [TOPS]**
- **Energy efficiency [TOPS/W]**
- **Area efficiency [TOPS/mm²]**

**ISLPED 2023**
International Symposium on Low Power Electronics and Design

**Performance**
- Increase of 7.5x

**Energy Efficiency**
- Increase of 53x

**Area Efficiency**
- Increase of 220x

**ITA System**
with 64 KIB SRAM
Future of ITA: Scaling up further

- Target workloads like GPT
- Floating-point capability

Accelerate LLMs and reach **100 TFLOPS** or higher!
A Primer on Hyper-Dimensional Computing

Mapping Symbols to Digital Representations
(The conventional Way)

Symbol Domain
(E.g. Letters, Numbers, Labels)

Representation Domain
(E.g. ASCII, 2s-complement, IEEE754 etc.)

Map

Symbol Domain:
- 1
- 2
- 3
- 42
- 0
- 255

Representation Domain:
- 0b000
- 0b011
- 0b111...

(E.g. Letters, Numbers, Labels)

(ASCII, 2s-complement, IEEE754 etc.)
A Primer on Hyper-Dimensional Computing

Mapping Symbols to Digital Representations
(The HDC Way)

Symbol Domain
(E.g. Letters, Numbers, Labels)

Map to
orthogonal Vectors

Representation Domain
(Binary Spatter Code, i.e. long Vectors of 0s and 1s)

D > 1000

e.g. 0100110101100…101110001

1 3 0

42

255

2 42

1
The properties of HD-Space:

Define Similarity Metric:
\[ V_1 \approx V_2 \iff d_{\text{Hamming}}(V_1, V_2) \ll D/2 \]

1. For large D, probability of two random vectors being similar to each other \( \approx 0 \)
   \( \rightarrow \) random vectors are quasi-orthogonal
A Primer on Hyper-Dimensional Computing

Mapping Symbols to Digital Representations
(The HDC Way)

Symbol Domain
(E.g. Letters, Numbers, Labels)

Map to orthogonal Vectors

Representation Domain
(Binary Spatter Code, i.e. long Vectors of 0s and 1s)

D>1000

- 3
- 0
- 1
- 42
- 2
- 255

E.g. 0100110101100…101110001
## A Primer on Hyper-Dimensional Computing

### Mapping Symbols to Digital Representations (The HDC Way)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>HD-Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>011010110100…..01001</td>
</tr>
<tr>
<td>2</td>
<td>110100100110…..11001</td>
</tr>
<tr>
<td>...</td>
<td>..........</td>
</tr>
<tr>
<td>255</td>
<td>101001101110.....00100</td>
</tr>
</tbody>
</table>

**Item Memory (IM)**

- **Symbol Domain** (E.g. Letters, Numbers, Labels)
- **Representation Domain** (Binary Spatter Code, i.e. long Vectors of 0s and 1s)

**Fixed Mapping using Randomly Selected Vectors**
A Primer on Hyper-Dimensional Computing

The properties of HD-Vectors:

Similarity Metric:
\[ V_1 \approx V_2 \iff d_{\text{Hamming}}(V_1, V_2) \ll D/2 \]

1. For large \( D \), probability of two random vectors being similar to each other \( \approx 0 \)
   \( \rightarrow \) random vectors are quasi-orthogonal

2. A noisy Vector can be recovered with high probability by looking up most-similar vector in IM
A Primer on Hyper-Dimensional Computing

**Item Memory (IM)**
Implemented as Content-Addresseable-Memory (CAM)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>HD-Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>011010110100.....01001</td>
</tr>
<tr>
<td>2</td>
<td>110100100110.....11001</td>
</tr>
<tr>
<td>...</td>
<td>........</td>
</tr>
<tr>
<td>255</td>
<td>101001101110.....00100</td>
</tr>
</tbody>
</table>
A Primer on Hyper-Dimensional Computing
A Primer on Hyper-Dimensional Computing

Low Dimensional Representation

Spatial Concentration of Information

biterror

42

complete loss of information

0 0 1 0 1 0 1 0 → 0 1 0 0 1 0 1 0

Representation Space Cardinality

1-to-1 mapping

Vector Symbolic Representation

biterror

42

information retained with very high probability

1 1 1 1 ··· 1 0 1 → 1 1 1 1 ··· 1 0 1

stochastic mapping
HDC Operators

Associative Lookup
- Search for most similar vector in IM
- Recover Item Vector from noisy Version

Binding (⨀)
- Take XOR of both vectors
- Output quasi-orthogonal to both operands
- Binding with same vector preserves similarity

Bundling (+)
- For each bit position, take majority of 0 or 1
- Output similar to all operands

Permutation (π)
- Permutate bit positions with fixed Permutation
- Unary Operation
- Maps vectors to orthogonal subspace
Building an ULP Accelerator for HDC

- Associative Memory
- Item Memory
- Flexible Control Path
- Bundling & Binding
Building an ULP Accelerator for HDC

- Associative Memory
- Item Memory
- Flexible Control Path
- Bundling & Binding
All-digital SCM-Based Associative Memory

• **Why digital?**
  - Technology-agnostic, easy integration in every digital SoC

• **Why a Standard-Cell Memory (SCM)?**
  - Supports weird aspect ratios (e.g. 32 x 1024 bit)
  - Natively support aggressive voltage scaling
  - Allows parallel access to all words if required (e.g. during min distance search)

  ![Near Memory!]
An Ephemeral Item Memory through Re-materialization

**Observation:** Storing many Item Vectors in a ROM is costly!

**Idea:** Use hardwired seed vector + random permutations

Still need many Permutations in Hardware (lots of wiring and multiplexing) 😞
An Ephemeral Item Memory through Re-materialization

**Observation:** Storing many Item Vectors in a ROM is costly!

**Idea:** Use hardwired seed vector + random permutations

- No ROM required
- Arbitrary number of Items supported
- Random Access to any Item Vector in $\log_2(n)$ cycles

Iterative Decomposition of the Permutation

**Iterative Decomposition of the Permutation**

- Low-dimensional Input
- $0b01101$
- Select
- $0100110101101001...001101$
Building an ULP Accelerator for HDC

- Associative Memory
- Item Memory
- Flexible Control Path
- Bundling & Binding
Hypnos – An Ultra-Low Power HDC Accelerator
Vega: A Ten-Core SoC for IoT Endnodes in GF22
Published in ISSCC21/JSSC Volume 57

Stage 1
(Single RV32)

Stage 2
(Heterogeneous Cluster)

Stage 0
(Always-on)

Conditional Wake Up

Tightly Coupled Data Memory

Logarithmic Interconnect

RISC-V core

Mem

I/$$

RISC-V core

I/$$

RISC-V core

I/$$

RISC-V core

I/$$

RISC-V core

I/$$

PULPissimo

Ext. Mem

Ext. ADC

Mem Cont

L2 Mem

I/O

Ext. Domain

Hypnos

I/O (SPI)

Preprocessor

Autonomous HD-Computing Unit

Always-on Domain

06/06/2023
Vega: A Ten-Core SoC for IoT Endnodes in GF22
Published in ISSCC21/JSSC Volume 57

Specifications

<table>
<thead>
<tr>
<th>Technology</th>
<th>GF22 UHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>670kGE</td>
</tr>
<tr>
<td>Max. Frequency</td>
<td>3 MHz</td>
</tr>
<tr>
<td>SCM-Memory</td>
<td>32 kBit</td>
</tr>
<tr>
<td>VDD</td>
<td>0.6V</td>
</tr>
</tbody>
</table>

3-channel HDC Inference Algorithm

<table>
<thead>
<tr>
<th>$f_{clk}$</th>
<th>32kHz</th>
<th>200kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>max. Sampling Rate</td>
<td>150 SPS/Channel</td>
<td>1kSPS/Channel</td>
</tr>
<tr>
<td>$P_{SWU, dynamic}$</td>
<td>0.99uW</td>
<td>6.21uW</td>
</tr>
<tr>
<td>$P_{SWU, leakage}$</td>
<td>0.7uW</td>
<td>0.7uW</td>
</tr>
<tr>
<td>$P_{SPI, dynamic}$</td>
<td>1.28uW</td>
<td>8.00uW</td>
</tr>
<tr>
<td>$P_{SWU, total}$</td>
<td>2.97uW</td>
<td>14.9uW</td>
</tr>
</tbody>
</table>
Vega: A Ten-Core SoC for IoT Endnodes in GF22
Published in ISSCC21/JSSC Volume 57

3000 µm

4000 µm

<table>
<thead>
<tr>
<th>Technology</th>
<th>22nm FDSOI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip Area</td>
<td>12mm²</td>
</tr>
<tr>
<td>SRAM</td>
<td>1728 kB</td>
</tr>
<tr>
<td>MRAM</td>
<td>4 MB</td>
</tr>
<tr>
<td>WU Sources</td>
<td>GPIO, RTC, Cognitive</td>
</tr>
<tr>
<td>VDD range</td>
<td>0.5V - 0.8V</td>
</tr>
<tr>
<td>VBB range</td>
<td>0V - +1.1V</td>
</tr>
<tr>
<td>Freq. Range</td>
<td>32 kHz - 450 MHz</td>
</tr>
<tr>
<td>Pow. Range</td>
<td>1.7 µW - 49.4 mW</td>
</tr>
<tr>
<td>Int. Perf.</td>
<td>15.6 GOPS</td>
</tr>
<tr>
<td>Int. Eff.</td>
<td>614 GOPS/W</td>
</tr>
<tr>
<td>F.P. Perf.</td>
<td>3.3 GFLOPS</td>
</tr>
<tr>
<td>F.P. Eff.</td>
<td>129 GFLOPS/W</td>
</tr>
<tr>
<td>Int. ML Perf.</td>
<td>32.2 GOPS</td>
</tr>
<tr>
<td>Int. ML Eff.</td>
<td>1.3 TOPS/W</td>
</tr>
</tbody>
</table>

Q&A