An In-Depth Learning of Matrix Multipliers for Deep Learning Accelerators

Presented by: Jose Yallouz, Intel Corporation

Joint work with: Amit Gradstein, Simon Rubanovich, Zeev Sperber, Adi Yoaz
“I have defined a "Matrix" as a rectangular array of terms, out of which different systems of determinants may be engendered as from the womb of a common parent.” Sylvester’s 1851 paper.

Matrix Theory Applications:
- Linear Algebra
- Graph Theory
- Physics
- Electronics
Matrix in Neural Networks

\[
\begin{bmatrix}
1 & 1 & 1 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
.5 & .5 & .5 \\
.5 & .5 & .5 \\
.5 & .5 & .5
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
1 & 2 & 3
\end{bmatrix}
\]

Go to Hidden Nodes

Bias
Node 1
Node 2
Node 3
\[
\begin{bmatrix}
1 & 1 & 1 \\
.5 & .5 & .5 \\
.5 & .5 & .5 \\
1 & 1 & 1
\end{bmatrix}
\cdot
\begin{bmatrix}
.3 & .1 \\
.5 & .15 \\
.5 & .15 \\
1 & .3
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
1 & 0 \\
1 & 0 \\
1 & 0
\end{bmatrix}
\]

Sigmoid Function
Weights
Output Layer
Sigmoid Function
Output
In-Datacenter Performance Analysis of a Tensor Processing Unit

Google, Inc., Mountain View, CA USA
jouppi@google.com

ABSTRACT

Many architects believe that major improvements in cost-energy-performance must now come from domain-specific hardware. This paper evaluates a custom ASIC—called a Tensor Processing Unit (TPU)—deployed in datacenters since 2015 that accelerates the inference phase of neural networks (NN). The heart of the TPU is a 65,536 8-bit MAC matrix multiply unit that offers a peak throughput of 92 TeraOps/second (TOPS) and a large (28 MiB) software-managed on-chip memory. The TPU’s deterministic execution model is a better match to the 99th-percentile response-time requirement of our NN applications than are the time-varying optimizations of CPUs and GPUs that help average throughput more than guaranteed latency. The lack of such features helps explain why, despite having myriad MACs and a big memory, the

KEYWORDS

DNN, MLP, CNN, RNN, LSTM, neural network, deep learning, domain-specific architecture, accelerator, TensorFlow, TPU, GPU

ACM Reference format:
https://doi.org/10.1145/3079856.3080246

1 INTRODUCTION TO NEURAL NETWORKS

The synergy between the large data sets in the cloud and the
Matrix Multiplication

\[ \hat{C}_{M \times N} = C_{M \times N} + A_{M \times K} \times B_{K \times N} = \]

\[
\begin{pmatrix}
\hat{c}_{11} & \cdots & \hat{c}_{1N} \\
\vdots & \ddots & \vdots \\
\hat{c}_{M1} & \cdots & \hat{c}_{MN}
\end{pmatrix} =
\begin{pmatrix}
c_{11} & \cdots & c_{1N} \\
\vdots & \ddots & \vdots \\
c_{M1} & \cdots & c_{MN}
\end{pmatrix} +
\begin{pmatrix}
a_{11} & \cdots & a_{1K} \\
\vdots & \ddots & \vdots \\
a_{M1} & \cdots & a_{MK}
\end{pmatrix} \times
\begin{pmatrix}
b_{11} & \cdots & b_{1N} \\
\vdots & \ddots & \vdots \\
b_{K1} & \cdots & b_{KN}
\end{pmatrix}
\]

\[ \hat{c}_{ij} = c_{ij} + \sum_{l=0}^{K} a_{ik} \cdot b_{kj} \]

The calculation of each element \( \hat{c}_{ij} \) requires \( K \) Multiply ACcumulate (MAC) operations.

Thus a matrix multiplication requires \( M \times K \times N \) MAC operations.

Matrix Multiplication is a high computing application!

- For \( M = K = N \), \( O(N^2) \) memory access results in \( O(N^3) \) operations.
Simple Implementation

- \( \hat{c}_{ij} = \sum_{l=0}^{K} a_{ik} \cdot b_{kj} = d_1 + d_2 + \ldots + d_t \)
- \( d_t = a_{i,2t-1} \cdot b_{2t-1,j} + a_{i,2t} \cdot b_{2t,j} \quad \forall 1 \leq t \leq \frac{n}{2} \)

\[ \frac{n}{2} \] CBs are required for \( n \times n \) matrix multiplication

Systolic array

\[
\begin{pmatrix}
    a_{11} & a_{12} & a_{13} \\
    a_{21} & a_{22} & a_{23} \\
    a_{31} & a_{32} & a_{33}
\end{pmatrix}
\times
\begin{pmatrix}
    b_{11} & b_{12} & b_{13} \\
    b_{21} & b_{22} & b_{23} \\
    b_{31} & b_{32} & b_{33}
\end{pmatrix}
\]
Systolic array

\[
\begin{pmatrix}
    a_{11} & a_{12} & a_{13} \\
    a_{21} & a_{22} & a_{23} \\
    a_{31} & a_{32} & a_{33}
\end{pmatrix}
\times
\begin{pmatrix}
    b_{11} & b_{12} & b_{13} \\
    b_{21} & b_{22} & b_{23} \\
    b_{31} & b_{32} & b_{33}
\end{pmatrix}
\]
Systolic array

\[
\begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{pmatrix} \times 
\begin{pmatrix}
b_{11} & b_{12} & b_{13} \\
b_{21} & b_{22} & b_{23} \\
b_{31} & b_{32} & b_{33}
\end{pmatrix}
\]
Systolic array

\[
\begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{pmatrix} \times \begin{pmatrix}
b_{11} & b_{12} & b_{13} \\
b_{21} & b_{22} & b_{23} \\
b_{31} & b_{32} & b_{33}
\end{pmatrix}
\]
Systolic array

\[
\begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{pmatrix}
\times
\begin{pmatrix}
b_{11} & b_{12} & b_{13} \\
b_{21} & b_{22} & b_{23} \\
b_{31} & b_{32} & b_{33}
\end{pmatrix}
\]

- After 3 cycles $C_{11}$ is ready
Systolic array

\[
\begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{pmatrix}
\times
\begin{pmatrix}
b_{11} & b_{12} & b_{13} \\
b_{21} & b_{22} & b_{23} \\
b_{31} & b_{32} & b_{33}
\end{pmatrix}
\]

- After 4 cycles \( C_{12} \) and \( C_{21} \) are ready

\[
\begin{align*}
a_{13} \cdot b_{33} \\
a_{23} \cdot b_{31} \\
a_{33}
\end{align*}
\]

\[
\begin{align*}
a_{12} \cdot b_{23} \\
a_{22} \cdot b_{22} \\
a_{32} \cdot b_{21}
\end{align*}
\]

\[
\begin{align*}
a_{11} \cdot b_{13} \\
a_{21} \cdot b_{12} \\
a_{31} \cdot b_{12}
\end{align*}
\]

CACT 2017
Systolic array

\[
\begin{pmatrix}
  a_{11} & a_{12} & a_{13} \\
  a_{21} & a_{22} & a_{23} \\
  a_{31} & a_{32} & a_{33}
\end{pmatrix}
\times
\begin{pmatrix}
  b_{11} & b_{12} & b_{13} \\
  b_{21} & b_{22} & b_{23} \\
  b_{31} & b_{32} & b_{33}
\end{pmatrix}
\]

• After 5 cycles $C_{13}$, $C_{22}$ and $C_{31}$ are ready
Systolic array

\[
\begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{pmatrix}
\times
\begin{pmatrix}
b_{11} & b_{12} & b_{13} \\
b_{21} & b_{22} & b_{23} \\
b_{31} & b_{32} & b_{33}
\end{pmatrix}
\]

• After 6 cycles $C_{23}$ and $C_{32}$ are ready
Systolic array

\[
\begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{pmatrix}
\times
\begin{pmatrix}
b_{11} & b_{12} & b_{13} \\
b_{21} & b_{22} & b_{23} \\
b_{31} & b_{32} & b_{33}
\end{pmatrix}
\]

- After 7 cycles $C_{33}$ is ready
- Computation Completed!
Deep Learning Accelerators Comparison

• Intel Nervana Neural Network Processor (NNP)
• Google Tensor Processing Unit (TPU)
• Nvidia Tesla Volta
• Movidius Myriad
• Mobileye EyeQ
## Deep Learning Accelerators Comparison

<table>
<thead>
<tr>
<th></th>
<th>Nervana</th>
<th>TPU (Gen2)</th>
<th>Volta</th>
<th>Myriad X</th>
<th>EyeQ5</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACs per unit</td>
<td>55 Tops</td>
<td>45 TFlops</td>
<td>120 TFlops</td>
<td>1Tops</td>
<td>24 Tops</td>
</tr>
<tr>
<td>Data type</td>
<td>flexpoint</td>
<td>FP16</td>
<td>FP16</td>
<td>FP16/ 8 bit fixed point</td>
<td></td>
</tr>
<tr>
<td>Power envelop</td>
<td>200W+</td>
<td>300W</td>
<td></td>
<td>10W</td>
<td></td>
</tr>
<tr>
<td>Technology</td>
<td>28nm</td>
<td>22nm</td>
<td>7nm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release Year</td>
<td>2017</td>
<td>2017</td>
<td>2018</td>
<td>2017</td>
<td>2020</td>
</tr>
<tr>
<td>Market</td>
<td>Server</td>
<td>Server</td>
<td>Server</td>
<td>End-user</td>
<td>End-user</td>
</tr>
</tbody>
</table>

CACT 2017
Is it a **Fair** Comparison?

The necessity of common benchmarks for Deep Learning (Inference/Training) performance analysis!

We should aim to create a comprehensive set of Deep Learning benchmarks (as SPEC for CPU) for performance analysis.

---

The necessity of common benchmarks for Deep Learning (Inference/Training) performance analysis!

We should aim to create a comprehensive set of Deep Learning benchmarks (as SPEC for CPU) for performance analysis.
Conclusions

• My personal passions:
  • Mathematics
  • Computer Science (CS)

• “As 150 years ago the math community adopts the Matrix theory, turning a *de facto* fundamental instrument in science, we are now adopting this important tool into Computer Science, specifically, into Computer Architecture”

• This adoption has been done through a specific high computing application:
  • Deep Learning

• “Is there any other killer application utilizing Matrix Multiplication?”

IAAI  (Intel Architecture for Artificial Intelligent)