Architectures for Compute-in-Memory
What’s a Deep Network?

Google DeepMind AlphaGo
What’s a Deep Network?

Image "Volvo XC90"
Why Should We Care?

From EE Times – September 27, 2016

"Today the job of training machine learning models is limited by compute, if we had faster processors we’d run bigger models...in practice we train on a reasonable subset of data that can finish in a matter of months. We could use improvements of several orders of magnitude – 100x or greater."

– Greg Diamos, Senior Researcher, SVAIL, Baidu
A Very Brief Introduction to Neural Nets

Convolutions account for more than 90% of overall computation, dominating runtime and energy consumption.
What is a Convolution?

Naïve 7-layer for-loop implementation:

\[
\text{for } (n=0; \ n<N; \ n++) \{ \\
\quad \text{for } (m=0; \ m<M; \ m++) \{ \\
\quad \quad \text{for } (x=0; \ x<F; \ x++) \{ \\
\quad \quad \quad \text{for } (y=0; \ y<E; \ y++) \{ \\
\quad \quad \quad \quad O[n][m][x][y] = B[m]; \\
\quad \quad \quad \quad \text{for } (i=0; \ i<R; \ i++) \{ \\
\quad \quad \quad \quad \quad \text{for } (j=0; \ j<S; \ j++) \{ \\
\quad \quad \quad \quad \quad \quad \text{for } (k=0; \ k<C; \ k++) \{ \\
\quad \quad \quad \quad \quad \quad \quad O[n][m][x][y] += I[n][k][Ux+i][Uy+j] \times W[m][k][i][j]; \\
\quad \quad \quad \quad \quad \} \\
\quad \quad \quad \quad \} \\
\quad \quad \quad \quad \} \\
\quad \quad \quad \quad O[n][m][x][y] = \text{Activation}(O[n][m][x][y]); \\
\quad \quad \} \\
\quad \} \\
\} \\
\} \\
\} \\
\]
What are our computations?

“deep” = multiple stages

multi-channel, multi-filter convolution
Central to neural networks is matrix multiplication
Matrix Multiplication

\[ C = AB \]

\[ c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \]

for \( i = 1 \) to \( n \)
  for \( j = 1 \) to \( n \)
    for \( k = 1 \) to \( n \)
      \[ c_{ij}^{k} = c_{ij}^{k-1} + a_{ik} b_{kj} \]
    end \( k \)
  end \( j \)
end \( i \)

for \( i = 1 \) to \( n \)
  for \( j = 1 \) to \( n \)
    for \( k = 1 \) to \( n \)
      \[ a( i, j, k ) = a_{ik}^{j} \]
      \[ b( i, j, k ) = b_{kj}^{i} \]
      \[ c( i, j, k ) = c_{ij}^{k} \]
    end \( k \)
  end \( j \)
end \( i \)
Systolic Matrix Multiplication

- Processors are arranged in a 2-D grid.
- Each processor accumulates one element of the product.
- The elements of the matrices to be multiplied are “pumped through” the array.
Systolic Matrix Multiplication

alignment in time

rows of a

columns of b

b_{2,2}

b_{2,1}

b_{2,0}

b_{1,2}

b_{1,1}

b_{1,0}

b_{0,2}

b_{0,1}

b_{0,0}

a_{0,2}

a_{0,1}

a_{0,0}

a_{1,2}

a_{1,1}

a_{1,0}

a_{2,2}

a_{2,1}

a_{2,0}

a_{1,2}\cdot b_{0,0}

+a_{0,1}\cdot b_{1,0}

+a_{0,2}\cdot b_{2,0}

a_{1,2}\cdot b_{0,0}

+a_{1,1}\cdot b_{1,0}

+a_{1,2}\cdot b_{2,0}

a_{2,2}\cdot b_{0,0}

+a_{2,1}\cdot b_{1,0}

+a_{2,2}\cdot b_{2,0}

a_{0,2}\cdot b_{0,1}

+a_{0,1}\cdot b_{1,1}

+a_{0,2}\cdot b_{2,1}

a_{1,2}\cdot b_{0,1}

+a_{1,1}\cdot b_{1,1}

+a_{1,2}\cdot b_{2,1}

a_{2,2}\cdot b_{0,1}

+a_{2,1}\cdot b_{1,1}

+a_{2,2}\cdot b_{2,1}

a_{0,2}\cdot b_{0,2}

+a_{0,1}\cdot b_{1,2}

+a_{0,2}\cdot b_{2,2}

a_{1,2}\cdot b_{0,2}

+a_{1,1}\cdot b_{1,2}

+a_{1,2}\cdot b_{2,2}

a_{2,2}\cdot b_{0,2}

+a_{2,1}\cdot b_{1,2}

+a_{2,2}\cdot b_{2,2}
Processing in Memory

```
<table>
<thead>
<tr>
<th>Operation</th>
<th>Energy (pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer ADD (8b)</td>
<td>0.03</td>
</tr>
<tr>
<td>Integer ADD (16b)</td>
<td>0.05</td>
</tr>
<tr>
<td>Integer ADD (32b)</td>
<td>0.1</td>
</tr>
<tr>
<td>Integer MULT (8b)</td>
<td>0.2</td>
</tr>
<tr>
<td>Integer MULT (32b)</td>
<td>3.1</td>
</tr>
<tr>
<td>8KB SRAM Read (32b)</td>
<td>5</td>
</tr>
<tr>
<td>32KB SRAM Read (32b)</td>
<td>10</td>
</tr>
<tr>
<td>1MB SRAM Read (32b)</td>
<td>50</td>
</tr>
</tbody>
</table>
```

"Computing’s Energy Problem (and what we can do about it)", M. Horowitz, ISSCC 2014
Processing in Memory

Digital/Analog processing in memory

Processing in memory can bring the combined energy of memory access and computation down to 50 fJ/Op
• data access energy and latency dominating
• data reuse and data compression
**digital**

- Memory
  - SRAM Bank
  - ALU / Digital processing

- data access energy and latency dominating
- data reuse and data compression

**near memory**

- Memory
  - SRAM Bank
  - Digital processing

- computation still digital
- eliminates data transfer costs
- memory read energy dominates
digital

- data access energy and latency dominating
- data reuse and data compression

near memory

- computation still digital
- eliminates data transfer costs
- memory read energy dominates

deep in-memory

- memory access and computation combined
- mixed signal computation
- significant energy & latency reduction
Standard Memory Design

- **energy** and **latency** costs of data access >> those of computation
- single row read per memory access
- column mux before SA
- $N_{col}/L$ bits per access
Standard Memory Design

\(2^n\) words of \(2^m\) bits each

If \(n \gg m\), fold by \(2^k\) into fewer rows of more columns

Good regularity – easy to design

Very high density if good cells are used
Cell size accounts for most of array size
  – Reduce cell size at expense of complexity

6T SRAM Cell
  – Used in most commercial chips
  – Data stored in cross-coupled inverters

Read:
  – Precharge bit, bit_b
  – Raise wordline

Write:
  – Drive data onto bit, bit_b
  – Raise wordline
Precharge both bitlines high
Then turn on wordline
One of the two bitlines will be pulled down by the cell
Ex: A = 0, A_b = 1
   - bit discharges, bit_b stays high
   - But A bumps up slightly

Read stability
   - A must not flip
   - N1 >> N2
Drive one bitline high, the other low
Then turn on wordline
Bitlines overpower cell with new value
Ex: $A = 0$, $A_b = 1$, bit = 1, bit_b = 0
  – Force $A_b$ low, then $A$ rises high

Writability
  – Must overpower feedback inverter
  – $N2 >> P1$
High bitlines must not overpower inverters during reads
But low bitlines must write new value into cell
Standard Memory Design

Read

Write
Standard Memory Design

- Cell size is critical: 26 x 45 \( \lambda \) (even smaller in industry)
- Tile cells sharing \( V_{\text{DD}} \), GND, bitline contacts

![Diagram of memory design](image)
- Precharge bitlines high before reads

- Equalize bitlines to minimize voltage difference when using sense amplifiers
- Clocked sense amp saves power
- Requires sense_clk after enough bitline swing
- Isolation transistors cut off large bitline capacitance
Compute-in-Memory Design

![Diagram of Compute-in-Memory Design]

- Precharge/Column Mux/Y-DEC
- Cross Bitline Processor
- Residual Digital Unit
- Inference/decisions
Compute-in-Memory Design

**Diagram:**
- Precharge/Column Mux/Y-DEC
- X-DEC
- BLP
- Cross Bitline Processor
- Residual Digital Unit
- Inference/decisions

**Highlighted Features:**
- Multi-row functional READ
  - Reads multiple-bits/row/precharge
- Analog, mixed-signal
  - Low-SNR processing
Compute-in-Memory Design

- Multi-row functional READ (reads multiple-bits/row/precharge)
- Analog, mixed-signal low-SNR processing
- Bitline processing (BLP) (SIMD analog processing)

Inference/decisions
Compute-in-Memory Design

1. Multi-row functional READ (reads multiple-bits/row/precharge)
2. Analog, mixed-signal low-SNR processing
3. Bitline processing (BLP) (SIMD analog processing)
4. Cross bitline processing (CBLP) (analog averaging enhances SNR)

Inference/decisions
Compute-in-Memory Design

1. Multi-row functional READ (reads multiple-bits/row/precharge)
2. Analog, mixed-signal low-SNR processing
3. Bitline processing (BLP) (SIMD analog processing)
4. Cross bitline processing (CBLP) (analog averaging enhances SNR)
Compute-in-Memory Design

1. Multi-row functional READ (reads multiple-bits/row/precharge)
2. Analog, mixed-signal low-SNR processing
3. Bitline processing (BLP) (SIMD analog processing)
4. Cross bitline processing (CBLP) (analog averaging enhances SNR)
5. Low complexity, low (decision) rate digital output
Compute-in-Memory Benefits?

<table>
<thead>
<tr>
<th>attribute</th>
<th>conventional</th>
<th>deep in-memory</th>
<th>benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>words read/access</td>
<td>$N_{col}/(LB_W)$</td>
<td>$N_{col}$</td>
<td>reduces # of memory accesses</td>
</tr>
<tr>
<td>BL swing/bit (mV)</td>
<td>250 – 300</td>
<td>20 – 150</td>
<td>reduces memory access energy</td>
</tr>
<tr>
<td># of rows/access</td>
<td>1</td>
<td>$B$</td>
<td>enhances throughput</td>
</tr>
</tbody>
</table>
Active Area of Research!

Adaboost; MNIST; energy savings = 13X; EDP reduction = 175X; [JSSC 2017, Princeton]

SVM, TM, k-NN, MF; MIT-CBCL, MNIST, ..; energy savings = 10X; EDP reduction = 50X; [JSSC 2018, UIUC]

RF with 64 trees; KUL traffic sign; energy savings = 3X EDP reduction = 7X [ESSCIRC 2017, UIUC]