Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators

Brandon Reagen
Paul Whatmough, Robert Adolf, Saketh Rama,
HK Lee, Sae Kyu Lee, Miguel Lobato
Gu-Yeon Wei, David Brooks

Harvard University
Machine Learning is ubiquitous
Machine Learning is ubiquitous
Machine Learning is ubiquitous

“What’s Minerva?”

“The goddess of wisdom”
Cut the cloud cord

“What’s Minerva?”

“The goddess of wisdom”
Still use cloud for training
Execute DNNs locally

Input data

Results

All-in-one
Execute DNNs locally

Local execution:
- Privacy
- Availability
- Comms cost
- Latency

Input data ➔ Results
Execute DNNs locally requires acceleration.
Requires going beyond today’s designs
Minerva: optimizing DNN accelerators

( Roman goddess of wisdom )
Minerva: optimizing DNN accelerators

( Roman goddess of wisdom )
Minerva: optimizing DNN accelerators

( Roman goddess of wisdom )
Minerva

<table>
<thead>
<tr>
<th>Baseline</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Select DNN</td>
<td></td>
</tr>
<tr>
<td>Accelerator uArch</td>
<td></td>
</tr>
</tbody>
</table>
Minerva

Baseline

- Select DNN
- Accelerator uArch

Optimizations
Minerva

Baseline

Select DNN

Accelerator uArch

Optimizations

Algorithm

Architecture

Circuit
## Minerva

<table>
<thead>
<tr>
<th>Baseline</th>
<th>Optimizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select DNN</td>
<td>Quantization</td>
</tr>
<tr>
<td>Accelerator uArch</td>
<td>Pruning</td>
</tr>
<tr>
<td></td>
<td>Fault mitigation</td>
</tr>
<tr>
<td></td>
<td>Baseline</td>
</tr>
<tr>
<td>-----</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td>Select DNN</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.6x</td>
</tr>
</tbody>
</table>
Minerva is not approximate computing

<table>
<thead>
<tr>
<th>Baseline</th>
<th>Optimizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select DNN</td>
<td>Accelerator uArch</td>
</tr>
<tr>
<td></td>
<td>Quantization</td>
</tr>
<tr>
<td></td>
<td>Pruning</td>
</tr>
<tr>
<td></td>
<td>Fault mitigation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Improvement</th>
<th>1.6x</th>
<th>1.9x</th>
<th>2.5x</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power (mW)</td>
<td>126</td>
<td>78</td>
<td>41</td>
</tr>
</tbody>
</table>

Error increase bounded by training noise
<table>
<thead>
<tr>
<th>Baseline</th>
<th>Optimizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select DNN</td>
<td>Accelerator uArch</td>
</tr>
<tr>
<td></td>
<td>Quantization</td>
</tr>
<tr>
<td></td>
<td>Pruning</td>
</tr>
<tr>
<td></td>
<td>Fault mitigation</td>
</tr>
</tbody>
</table>
Fully connected Deep Neural Networks
Fully connected Deep Neural Networks

Network layer
Fully connected Deep Neural Networks

\[ \begin{align*}
    x_0 \\
    x_1 \\
    x_2
\end{align*} \]
Fully connected Deep Neural Networks

Input  \downarrow

\[ x_0 \xrightarrow{w_0} \]

\[ x_1 \xrightarrow{w_1} \]

\[ x_2 \xrightarrow{w_2} \]

Neuron

Weight
Fully connected Deep Neural Networks

\[ x_0 * w_0 + x_1 * w_1 + x_2 * w_2 \]
Fully connected Deep Neural Networks

Neuron output is its activity

\[ x_0 \cdot w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 \]
Fully connected Deep Neural Networks
Fully connected Deep Neural Networks
Fully connected Deep Neural Networks

Output layer

Classification

x_0 \rightarrow n_0 \rightarrow m_0 \rightarrow 0.1

x_1 \rightarrow n_1 \rightarrow m_1 \rightarrow 0.8

x_2 \rightarrow n_2 \rightarrow m_2 \rightarrow 0.1

\text{Input layer}

\text{Hidden layers}

\text{Output layer}

ClassificaRon
Choosing a DNN
Choosing a DNN

Neurons per layer
Choosing a DNN

Number of layer

Neurons per layer

Choosing a DNN

Number of layer
DNN training space

![Graph showing prediction error vs total number of DNN weights]

- Prediction Error (%) on the y-axis
- Total number of DNN weights on the x-axis
- The graph illustrates the relationship between prediction error and the total number of DNN weights.
<table>
<thead>
<tr>
<th>Baseline</th>
<th>Optimizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select DNN</td>
<td>Quantization</td>
</tr>
<tr>
<td></td>
<td>Pruning</td>
</tr>
<tr>
<td>3 layers</td>
<td>Fault</td>
</tr>
<tr>
<td>256 nodes</td>
<td>mitigation</td>
</tr>
<tr>
<td>1.4% error</td>
<td></td>
</tr>
</tbody>
</table>
Mapping DNNs to gates
Mapping DNNs to gates
Design of a datapath lane
Design of a datapath lane
Design of a datapath lane
Design of a datapath lane
Design of a datapath lane

Activity SRAM

Weight SRAM
Design of a datapath lane
Accelerator architecture

Weight Activity SRAMs

Datapath lanes

$X_0$, $X_1$, $X_2$, $X_3$, $X_4$
Accelerator architecture

Weight
Activity
SRAMs

Datapath lanes

X₀
X₁
X₂
X₃
X₄
Accelerator architecture

Weight Activity SRAMs

Datapath lanes

x_0 x_1 x_2 x_3 x_4
Accelerator architecture

- Weight Activity SRAMs
- Datapath lanes
Accelerator architecture

Weight Activity SRAMs

Datapath lanes
Accelerator architecture

Datapath lanes

Weight
Activity
SRAMs
Selecting an accelerator implementation

Weight and Activity SRAMs
Selecting an accelerator implementation

Weight and Activity
SRAMs

Memory Bandwidth
Selecting an accelerator implementation

Weight and Activity
SRAMs

Memory Bandwidth

Lane parallelism
Selecting an accelerator implementation

Weight and Activity
SRAMs

Lane parallelism

Memory Bandwidth

Number of lanes
Accelerator microarchitecture design space
<table>
<thead>
<tr>
<th>Baseline</th>
<th>Optimizations</th>
<th>Improvement</th>
<th>Power (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select DNN</td>
<td>Accelerator uArch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 layers 256 nodes</td>
<td>250 Mhz 12K pred/sec</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Quantization</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pruning</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fault mitigation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Improvement</td>
<td>126</td>
</tr>
</tbody>
</table>
Datapath lane
Opportunity for heterogeneous datatypes
Datatype selection

Weights

Activities

Products
Aggressive datatype quantization

Weights

Activities

Products
Aggressive datatype quantization

Weights

Activities

Products

10bits

Standard

Minerva

Standard
Aggressive datatype quantization

Weights

- Standard
- Minerva

Activities

- Standard
- Minerva

Products

- Standard
- Minerva

10bits
## Minerva

<table>
<thead>
<tr>
<th>Baseline</th>
<th>Optimizations</th>
<th>Power (mW)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select DNN</td>
<td>Accelerator uArch</td>
<td>126</td>
<td>1.6x</td>
</tr>
<tr>
<td>3 layers</td>
<td>250 Mhz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256 nodes</td>
<td>12K pred/sec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quantization</td>
<td>Pruning</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>Weight = 8b</td>
<td>Weight = 8b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activ = 6b</td>
<td>Activ = 6b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prod = 9b</td>
<td>Prod = 9b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fault mitigation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DNN amenability to pruning
DNN amenability to pruning

![Histogram showing count (in millions) of activity values, with a peak at Zeros (7 millions)].
DNN amenability to pruning

![Graph showing DNN amenability to pruning]

- **Zeros**: Approximately 7 million
- **Small non-zeros**: Approximately 3 million
Predicating execution in the datapath lane

Activity SRAM

Weight SRAM

Enable
Pruned ops using threshold
Pruned ops using threshold

75% operation elided
## Minerva

<table>
<thead>
<tr>
<th>Baseline</th>
<th>Optimizations</th>
<th>Fault mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select DNN</td>
<td>Accelerator uArch</td>
<td>Quantization</td>
</tr>
<tr>
<td>3 layers 256 nodes</td>
<td>250 Mhz 12K pred/sec</td>
<td>Weight = 8b Activ = 6b Prod = 9b</td>
</tr>
</tbody>
</table>

### Improvement

<table>
<thead>
<tr>
<th>Improvement</th>
<th>1.6x</th>
<th>1.9x</th>
</tr>
</thead>
</table>

### Power (mW)

| Power (mW) | 126 | 78 | 41 |
The potential and problems with reducing SRAM supply voltage
The potential and problems with reducing SRAM supply voltage
Fault mitigation

No protection

Word masking

Bit masking
Fault mitigation

No protection

Word masking

Bit masking
Fault mitigation

No protection

Word masking

Bit masking

Error

Fault probability

(low) 0.01% (high)
Word masking example

Stored weight

1 0 1 1 0 0 1 0
Word masking example

<table>
<thead>
<tr>
<th>Stored weight</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Read weight</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

"X" indicates the masked word.
**Word masking example**

<table>
<thead>
<tr>
<th>Stored weight</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read weight</td>
<td>1</td>
<td>0</td>
<td>×</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Weight used</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Fault mitigation

No protection

Word masking

Bit masking

Fault probability

(low)  (high)
Bit masking example

Stored weight

```
1 0 1 1 0 0 1 0
```
Bit masking example

<table>
<thead>
<tr>
<th>Stored weight</th>
<th>Read weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 1 0 0 1 0</td>
<td>1 0 X 1 X 0 1 0</td>
</tr>
</tbody>
</table>
Bit masking example

<table>
<thead>
<tr>
<th>Stored weight</th>
<th>Read weight</th>
<th>Weight used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 1 0 0 1 0</td>
<td>1 0 1 1 0 0 1 0</td>
<td>1 0 1 1 1 1 0 1 0</td>
</tr>
</tbody>
</table>

Round faulty bits towards zero by copying the sign-bit.
Fault mitigation

No protection

Word masking

Bit masking

Fault probability

(low) 0.01% (high) 4.4%

Error

Error

Error
Fault mitigation

No protection

Word masking

Bit masking
Reduces supply by >200mV
## Minerva

<table>
<thead>
<tr>
<th>Baseline</th>
<th>Optimizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select DNN</td>
<td>Accelerator uArch</td>
</tr>
<tr>
<td>3 layers</td>
<td>250 Mhz</td>
</tr>
<tr>
<td>256 nodes</td>
<td>12K pred/sec</td>
</tr>
<tr>
<td></td>
<td>Quantization</td>
</tr>
<tr>
<td></td>
<td>Weight = 8b</td>
</tr>
<tr>
<td></td>
<td>Activ = 6b</td>
</tr>
<tr>
<td></td>
<td>Prod = 9b</td>
</tr>
<tr>
<td></td>
<td>Pruning</td>
</tr>
<tr>
<td></td>
<td>&gt;75% ops elided</td>
</tr>
<tr>
<td></td>
<td>Fault mitigation</td>
</tr>
<tr>
<td></td>
<td>4.4% Fault rate</td>
</tr>
</tbody>
</table>

### Improvement
- 1.6x
- 1.9x
- 2.5x

### Power (mW)
- Baseline: 126 mW
- Optimizations: 78 mW
- Fault mitigation: 41 mW
- 8x power reduction
See paper for more results

Generalizes to other datasets
  – MNIST, Reuters, WebKB, Forest, 20NG
  – 8.1x average savings

Consider programmable DNN engine
  – Increases power by 1.4x

Validated against PnR’d RTL
  – 12% simulation error
Taped out Minerva prototype

Last week!

28nm TSMC
Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators

Minerva Optimizations

- Quantization
- Pruning
- Fault mitigation

Minerva design

8x less power
Same model accuracy

Baseline

Harvard Robobee
Backup
The real Minerva design flow
Error bounds

Intrinsic Error Variation

Prediction Error (%) vs. Epoch

Max

Mean

+1 σ

-1 σ

Min
ConvNets are not all there is.

<table>
<thead>
<tr>
<th></th>
<th>Add</th>
<th>Sub</th>
<th>Mul</th>
<th>Div</th>
<th>Pow</th>
<th>Softmax</th>
<th>MatMul</th>
<th>CrossEntropy</th>
<th>MaxPoolGrad</th>
<th>Sum</th>
<th>Conv2D</th>
<th>Conv2DBackFilter</th>
<th>Conv2DBackInput</th>
<th>RandomNormal</th>
<th>ApplyAdam</th>
<th>ApplyRMSProp</th>
<th>Transpose</th>
<th>Tile</th>
<th>Select</th>
<th>Pad</th>
<th>Reshape</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>seq2seq</td>
<td>3</td>
<td>2</td>
<td>35</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>memn2n</td>
<td>2</td>
<td>1</td>
<td>33</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>12</td>
<td>2</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>9</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>speech</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>89</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>autoencoder</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>58</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>8</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>residual</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>vgg</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>alexnet</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>31</td>
<td>26</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>atari</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>33</td>
<td>27</td>
<td>20</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Validation and chip prototyping

<table>
<thead>
<tr>
<th></th>
<th>Aladdin</th>
<th>Layout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictions/s</td>
<td>11,820</td>
<td></td>
</tr>
<tr>
<td>$\mu$J/Prediction</td>
<td>1.38</td>
<td></td>
</tr>
<tr>
<td>1 Lane (mm$^2$)</td>
<td>0.12</td>
<td>0.12</td>
</tr>
<tr>
<td>16 Lanes (mm$^2$)</td>
<td>1.84</td>
<td>1.86</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>16.3</td>
<td>18.5</td>
</tr>
</tbody>
</table>

<12% error
Compared to others