ComPEND: Computation Pruning through Early Negative Detection for ReLU in a Deep Neural Network Accelerator

June 13, 2018

Dongwoo Lee, Sungbum Kang, Kiyoung Choi

Neural Processing Research Center (NPRC)
Seoul National University
Outline

• Motivation
• Early Negative Detection (END)
• Computation Pruning thru END (ComPEND)
• Evaluation
• Conclusion
Motivation

- Perceptron

$$x = \sum_{i=1}^{N} A_i \times W_i$$

- Rectified linear unit (ReLU, \( f(x) = \max(0, x) \)) is widely used as an activation function for DNN.
Motivation

• Perceptron

\[ x = \sum_{i=1}^{N} A_i \times W_i \]

• Rectified linear unit (ReLU, \( f(x) = \max(0, x) \)) is widely used as an activation function for DNN.
Motivation

- Perceptron

Rectified linear unit (ReLU, \(f(x) = \max(0, x)\)) is widely used as an activation function for DNN.

If we know \textit{a priori} that \(x \leq 0\), we can skip unnecessary computations and simply set ReLU output to zero.
Motivation

- Distribution of negative inputs to ReLU functions in VGG-16
  - More than 60%
Early Negative Detection (END)

• Two’s complement number representation (4 bits)

<table>
<thead>
<tr>
<th>Negative</th>
<th>Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111 = -8+7 = -1</td>
<td>0111</td>
</tr>
<tr>
<td>1110 = -8+6 = -2</td>
<td>0110</td>
</tr>
<tr>
<td>1101 = -8+5 = -3</td>
<td>0101</td>
</tr>
<tr>
<td>1100 = -8+4 = -4</td>
<td>0100</td>
</tr>
<tr>
<td>1011</td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>0100</td>
<td>0011</td>
</tr>
<tr>
<td>0101</td>
<td>0010</td>
</tr>
<tr>
<td>0110</td>
<td>0001 = -0+1 = +1</td>
</tr>
<tr>
<td>0111</td>
<td>0000 = -0+0 = +0</td>
</tr>
</tbody>
</table>

For a B-bit number $W : (w_{B-1} \ w_{B-2} \ w_{B-3} \ldots \ w_1 \ w_0)$

$$W = w_{B-1} \times (-2^{B-1}) + \sum_{k=0}^{B-2} w_k \times (+2^k)$$
Early Negative Detection (END)

• Inverted two’s complement number representation (4 bits)

Positive

- 1111 = +8-7 = +1
- 1110 = +8-6 = +2
- 1101 = +8-5 = +3
- 1100 = +8-4 = +4
- 1011
- 1010
- 1001
- 1000

Negative

- 0111
- 0110
- 0101
- 0100
- 0011
- 0010
- 0001 = +0-1 = -1
- 0000 = +0-0 = -0

For a B-bit number $W : (w_{B-1} w_{B-2} w_{B-3} \ldots w_1 w_0)$

$W = w_{B-1} \times (+2^{B-1}) + \sum_{k=0}^{B-2} w_k \times (-2^k)$
Early Negative Detection (END)

- Inverted two’s complement representation for negative detection

\[
\begin{array}{c|c|c}
\text{Decimal} & \text{2’s complement} \\
\hline
\text{Activation: } 5 & 0 & 1 \\
\text{Weight: } -6 & 0 & 1 \\
-3 & 0 & 1 \\
\end{array}
\]

- \( \text{ReLU} \) activation function
- \( 0 \) output
Early Negative Detection (END)

- Inverted two’s complement representation for negative detection

<table>
<thead>
<tr>
<th>Decimal</th>
<th>2’s complement</th>
<th>Inverted 2’s complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activation: 5</td>
<td>0 1 0 1</td>
<td>0 1 0 1</td>
</tr>
<tr>
<td>Weight: -6</td>
<td>1 0 1 0</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>- 3 0</td>
<td>+ 0 0 0 1</td>
<td>+ 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>+ 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 1 1 0 0 0 1 0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

ReLU
Early Negative Detection (END)

- Inverted two’s complement representation for negative detection

<table>
<thead>
<tr>
<th>Activation:</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight:</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>- 3 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decimal</th>
<th>2’s complement</th>
<th>Inverted 2’s complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{ReLU} & \rightarrow 0
\end{align*}
\]
Early Negative Detection (END)

- Inverted two’s complement representation for negative detection

<table>
<thead>
<tr>
<th>Decimal</th>
<th>2’s complement</th>
<th>Inverted 2’s complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1 0 1 0</td>
<td>0 1 0 1</td>
</tr>
<tr>
<td>-6</td>
<td>0 1 0 1</td>
<td>0 1 0 1</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{Activation:} & \quad 5 \\
\text{Weight:} & \quad -6 \\
\end{align*}
\]

\[
\begin{align*}
&\quad \begin{array}{c} 1 \end{array} \quad \begin{array}{c} 0 \end{array} \quad \begin{array}{c} 1 \end{array} \\
+ & \quad \begin{array}{c} 0 \end{array} \quad \begin{array}{c} 0 \end{array} \quad \begin{array}{c} 0 \end{array} \\
\hline
\text{ReLU:} & \quad \begin{array}{c} 1 \end{array} \quad \begin{array}{c} 1 \end{array} \quad \begin{array}{c} 1 \end{array} \quad \begin{array}{c} 0 \end{array} \quad \begin{array}{c} 0 \end{array} \quad \begin{array}{c} 1 \end{array} \quad \begin{array}{c} 0 \end{array} \quad \begin{array}{c} 1 \end{array} \quad \begin{array}{c} 1 \end{array} \quad \begin{array}{c} 0 \end{array} \quad \begin{array}{c} 0 \end{array} \quad \begin{array}{c} 0 \end{array} \quad \begin{array}{c} 0 \end{array}
\end{align*}
\]

\[\text{Skipped!}\]
Early Negative Detection (END)

- Two’s complement representation

\[ W = w_{B-1} \times (-2^{B-1}) + \sum_{k=0}^{B-2} w_k \times (+2^k) \]

- Inverted two’s complement representation

\[ W = w_{B-1} \times (+2^{B-1}) + \sum_{k=0}^{B-2} w_k \times (-2^k) \]

Stop here!
Early Negative Detection (END)

- For multiple inputs

\[
x = \sum_{i=1}^{N} A_i \times W_i = A_1 \times [w_{1,B-1} \times 2^{B-1} - w_{1,B-2} \times 2^{B-2} - w_{1,B-3} \times 2^{B-3} - \cdots ] \\
+ A_2 \times [w_{2,B-1} \times 2^{B-1} - w_{2,B-2} \times 2^{B-2} - w_{2,B-3} \times 2^{B-3} - \cdots ] \\
+ A_N \times [w_{N,B-1} \times 2^{B-1} - w_{N,B-2} \times 2^{B-2} - w_{N,B-3} \times 2^{B-3} - \cdots ]
\]
Early Negative Detection (END)

- For multiple inputs

\[
x = \sum_{i=1}^{N} A_i \times W_i = A_1 \times [w_{1,B-1} \times 2^{B-1} - w_{1,B-2} \times 2^{B-2} - w_{1,B-3} \times 2^{B-3} - \ldots ] + A_2 \times [w_{2,B-1} \times 2^{B-1} - w_{2,B-2} \times 2^{B-2} - w_{2,B-3} \times 2^{B-3} - \ldots ] + \ldots + A_N \times [w_{N,B-1} \times 2^{B-1} - w_{N,B-2} \times 2^{B-2} - w_{N,B-3} \times 2^{B-3} - \ldots ]
\]
Early Negative Detection (END)

• For multiple inputs

\[ x = \sum_{i=1}^{N} A_i \times W_i = A_1 \times [w_{1,B-1} \times 2^{B-1} - w_{1,B-2} \times 2^{B-2} - w_{1,B-3} \times 2^{B-3} - \ldots ] + A_2 \times [w_{2,B-1} \times 2^{B-1} - w_{2,B-2} \times 2^{B-2} - w_{2,B-3} \times 2^{B-3} - \ldots ] + A_N \times [w_{N,B-1} \times 2^{B-1} - w_{N,B-2} \times 2^{B-2} - w_{N,B-3} \times 2^{B-3} - \ldots ] \]
Computation Pruning thru END (ComPEND)

- Bit-serial sum of products
  - Takes multiple steps, but the area of a bit-serial unit is much smaller
  - Can integrate more units → higher performance
  - Similar to Stripes (P. Judd et al., MICRO 2016)

< Conventional sum of products >

< Bit-serial sum of products >
Computation Pruning thru END (ComPEND)

• Overall architecture of ComPEND

- 9x16 array of PUs
- 32 16-bit inputs per PU
- 9x16x32 inputs at a time (3x3x512 filter)
- 16 + 1 additional PUs
Computation Pruning thru END (ComPEND)

- **DATA packing**
  - Input activation block
    - 32 activations of same X, Y
      - \( A_{1,1,1}, A_{1,1,2}, A_{1,1,3}, A_{1,1,4}, \ldots, A_{1,1,31}, A_{1,1,32} \)
      - 16-bit
      - 512-bit

- **Weight bits block**
  - 512 bits of weights in same bit position
    - \( W_{1,1,1}, W_{1,1,2}, W_{1,1,3}, W_{1,1,4}, \ldots, W_{1,1,511}, W_{1,1,512} \)
      - MSB
      - MSB-1
      - LSB
      - 1-bit
      - 512-bit

< in the case of \( F_z = 512 \) >
Computation Pruning thru END (ComPEND)

- Processing unit

- 32-input 16-bit adder tree
- 32 16-bit input activation registers
- 32-bit weight bits register
Computation Pruning thru END (ComPEND)

- **Memory controller**
  - Manages all kinds of memory-involved data transfers

- **Weight blocks**
  - Off-chip memory -> STT-RAM
  - STT-RAM -> Weight Buffers (WBs)
  - WBs -> Weight registers in PUs

- **Activation blocks**
  - Off-chip memory -> Activation Buffers (ABs)
  - Off-chip memory -> Registers in PUs
    (FC layers: activation blocks are moved directly from off-chip memory to registers)
  - ABs -> Registers in PUs

- **Output activation blocks**
  - Global controller -> Off-chip memory
Computation Pruning thru END (ComPEND)

• Provider network

\[ \begin{array}{cccc}
A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\
A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\
A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\
\end{array} \]

\[ \begin{array}{cccc}
A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} \\
A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} \\
A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} \\
\end{array} \]

Sliding window

\[ \begin{array}{cccc}
W_{1,1} & W_{1,2} & W_{1,3} & \\
\vdots & \vdots & \vdots & \\
W_{3,3} & & & \\
\end{array} \]

< Connection type 1 >

\[ \begin{array}{cccc}
W_{1,1} & \rightarrow & A_{1,1} \\
W_{1,2} & \rightarrow & A_{1,2} \\
W_{1,3} & \rightarrow & A_{1,3} \\
W_{3,3} & \rightarrow & A_{3,3} \\
\end{array} \]

< Connection type 2 >

• Inputs: 32 x 9 x 16 bits
• outputs: 32 x 9 x 16 bits
• Activation reuse in PUs
  • During 2D convolution with 3x3 filters
  • Reconfiguration with 9 types of connections for shuffling weights
Computation Pruning thru END (ComPEND)

• Global controller

• 16 decision units
  • Decides final sum of products
    • Zero if DATA is negative
    • DATA if last position is LSB

• Pipeline list
  • id: filter ID
  • pos: bit position in 16-bit weights
  • head: current output of adder tree

• Entry board
  • id: filter ID
  • last pos: last position in the pipeline
  • DATA: partial sum
Computation Pruning thru END (ComPEND)

- Global controller

Filling up the pipeline

P1: The next bit in the bit-serial computation

P2: A new sum of products that has not yet been entered into the pipeline

P3: The next step of a sum of products whose prior step is still in the pipeline

\[
F_p: (w_{i,B-1}, w_{i,B-2}, \ldots, w_{i,1}, w_{i,0})
\]

\[
F_q: (w_{j,B-1}, w_{j,B-2}, \ldots, w_{j,1}, w_{j,0})
\]
Computation Pruning thru END (ComPEND)

• Operation pipeline

1. Weight buffers ->
2. Provider network (2) ->
3. Processing unit array (3) ->
4. Global controller
Evaluation

• Pre-trained weights of VGG-16 network and 1000 images from ImageNet ILSVRC-2012
• In-house cycle-accurate timing simulator by using C++ with DRAMSim2 for off-chip memory
• CACTI 6.5 to model SRAM
• NVSim for on-chip STT-RAM
• Synopsys Design Compiler with TSMC 45nm technology library with 0.9V to get parameters of timing/power/area for PUs and Provider Network
Evaluation

• VGG-16 network

<table>
<thead>
<tr>
<th>Layer</th>
<th>Imap</th>
<th>filter</th>
<th># of filter</th>
<th>Omap</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>224x224x3</td>
<td>3x3x3</td>
<td>64</td>
<td>224x224x64</td>
</tr>
<tr>
<td>C2</td>
<td>224x224x64</td>
<td>3x3x64</td>
<td>64</td>
<td>224x224x64</td>
</tr>
<tr>
<td>C3</td>
<td>112x112x64</td>
<td>3x3x64</td>
<td>128</td>
<td>112x112x128</td>
</tr>
<tr>
<td>C4</td>
<td>112x112x128</td>
<td>3x3x128</td>
<td>128</td>
<td>112x112x128</td>
</tr>
<tr>
<td>C5</td>
<td>56x56x128</td>
<td>3x3x128</td>
<td>256</td>
<td>56x56x256</td>
</tr>
<tr>
<td>C6</td>
<td>56x56x256</td>
<td>3x3x256</td>
<td>256</td>
<td>56x56x256</td>
</tr>
<tr>
<td>C7</td>
<td>56x56x256</td>
<td>3x3x256</td>
<td>256</td>
<td>56x56x256</td>
</tr>
<tr>
<td>C8</td>
<td>28x28x256</td>
<td>3x3x256</td>
<td>512</td>
<td>28x28x512</td>
</tr>
<tr>
<td>C9</td>
<td>28x28x512</td>
<td>3x3x512</td>
<td>512</td>
<td>28x28x512</td>
</tr>
<tr>
<td>C10</td>
<td>28x28x512</td>
<td>3x3x512</td>
<td>512</td>
<td>28x28x512</td>
</tr>
<tr>
<td>C11</td>
<td>14x14x512</td>
<td>3x3x512</td>
<td>512</td>
<td>14x14x512</td>
</tr>
<tr>
<td>C12</td>
<td>14x14x512</td>
<td>3x3x512</td>
<td>512</td>
<td>14x14x512</td>
</tr>
<tr>
<td>C13</td>
<td>14x14x512</td>
<td>3x3x512</td>
<td>512</td>
<td>14x14x512</td>
</tr>
<tr>
<td>F1</td>
<td>7x7x512</td>
<td>7x7x512</td>
<td>4096</td>
<td>1x1x4096</td>
</tr>
<tr>
<td>F2</td>
<td>1x1x4096</td>
<td>1x1x4096</td>
<td>4096</td>
<td>1x1x4096</td>
</tr>
<tr>
<td>F3</td>
<td>1x1x4096</td>
<td>1x1x4096</td>
<td>1000</td>
<td>1x1x1000</td>
</tr>
</tbody>
</table>

• We use 15 layers in the VGG-16 network as workloads, excluding layer F1.
• F1 is excluded since the total size of input activations is too big.
• Inputs to C1 are raw data that can be negative.
  ➔ The pruning scheme cannot be applied.
  ➔ C1 is implemented without ComPEND.
Evaluation

• Configuration

<table>
<thead>
<tr>
<th>Frequency</th>
<th>1 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing unit</td>
<td>Total 161 PUs (9 x 16 + 16 + 1), 2 bytes 32-input PU, 1 cycle for a PU, 3 pipeline stages for a PU array</td>
</tr>
<tr>
<td>Peak throughput</td>
<td>Total 4.608 TOPS or 288 GMACS</td>
</tr>
<tr>
<td>Weight buffer</td>
<td>Total 288 Kbytes, 9 x 32 Kbytes SRAM, 64 bytes, 1 cycle for a 32 Kbytes WB</td>
</tr>
<tr>
<td>Activation buffer</td>
<td>Total 128 Kbytes, 4 x 32 Kbytes SRAM, 64 bytes, 1 cycle for a 32 Kbytes AB</td>
</tr>
<tr>
<td>Provider network</td>
<td>9 x 64 bytes input, 9 x 64 bytes output, 1 cycle for a stage, 2 pipeline stages</td>
</tr>
<tr>
<td>Global controller</td>
<td>16 decision units, 1 cycle for each unit</td>
</tr>
<tr>
<td>STT-RAM</td>
<td>Total 4.5 Mbytes, 9 x 512 Kbytes STT-RAM, 64 bytes, 2 cycles for read, 11 cycles for write of a 512 Kbytes STT-RAM</td>
</tr>
<tr>
<td>DRAM</td>
<td>16 Gbytes, 8 Banks, 1333 MHz (DDR3_micron_32M_8B_x8_sg15)</td>
</tr>
</tbody>
</table>

• Area

<table>
<thead>
<tr>
<th></th>
<th>Area (mm²)</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUs</td>
<td>0.75</td>
<td>13%</td>
</tr>
<tr>
<td>Provider</td>
<td>0.09</td>
<td>2%</td>
</tr>
<tr>
<td>WBs</td>
<td>1.27</td>
<td>23%</td>
</tr>
<tr>
<td>ABs</td>
<td>0.56</td>
<td>10%</td>
</tr>
<tr>
<td>STT-RAM</td>
<td>2.93</td>
<td>52%</td>
</tr>
<tr>
<td>Total</td>
<td>5.62</td>
<td>100%</td>
</tr>
</tbody>
</table>

※ Peak throughput (32-input PU × 16 PUs in a row × 9 rows × 1GHz = 4.6 TOPS)
Evaluation

• Runtime
  • Reduced by 16.62% on average compared to that without ComPEND for 15 layers

- MEM_STT: reads/writes between off-chip memory and STT-RAM
- STT_WB: runtime of reads/writes between STT-RAM and WB
- MEM_WB: reads/writes between off-chip memory and WB
- MEM_AB: reads/writes between off-chip memory and AB
- AB_PU: reads/writes between AB and registers in PUs
- RUN_PU: computation in PUs

< for VGG-16 layers >
Evaluation

- Energy (dynamic & static) consumption
  - Reduced by 23.50% on average for 15 layers

- D/S_CTRL: global controller
- D/S_NET: provider network
- D/S_STT: STT-RAM
- D/S_AB: activation buffers
- D/S_WB: weight buffer
- D/S_PU: processing units

Left bars: without ComPEND
Right bars: with ComPEND  < for VGG-16 layers >
Evaluation

• Power consumption
  • Average over 15 layers
    • Without ComPEND: 1.12 Watt
    • With ComPEND: 1.03 Watt

< for VGG-16 layers >
Evaluation

• Energy-delay product
  • ComPEND reduces EDP and ED$^2$P by 36.21% and 46.81% for the execution of the 15 layers

< for VGG-16 layers >
Conclusion

• Proposed the concept of END (early negative detection) based on inverted two’s complement

• Proposed an architecture that implements ComPEND

• Achieved 16.62% higher speed and 23.50% less energy consumption for inference

• Future work
  • Combining with other zero-skipping approaches
  • Handling layers (say, F1 in VGG-16) exceeding the capacity of the architecture
THANK YOU