Efficient Implementations of Deep Neural Network Hardware

Takashi Miyamori, General Manager
Center for Semiconductor Research & Development

Toshiba Electronic Devices & Storage Corporation
Deep Learning Everywhere

Cloud

Data Center

> MW

Internet

Face Recognition for Door Security

Bottleneck on communication

SoC < 3~5W

Realtime processing

SoC

© 2015 Toshiba Corporation
Semantic Segmentation

• Classify objects in each pixel

**Input Image**

![Input Image](image)

**DNN**

**Result of Pixel Segmentation**

![Result of Pixel Segmentation](image)

**e.g. SegNet**

**Image X**

Pooling

Up sampling

Classifier

Semantic Labels

y

Road Detection by DNN
Outline

• Technology Trends

• LOGNET: energy-efficient neural networks using logarithmic computation (Stanford Univ. & Toshiba) [ICASSP 2017]

• TDNN: Time-Domain Neural Network (Toshiba) [A-SSCC 2016]
Efficient DNN Implementations

• Improvement of Network Models
  – GoogLeNet, ResNet

• Reduction of Parameters (# of data, bit width) and Compression
  – Deep Compression (Stanford): Pruning, Quantization, Huffman Coding
  – Binarized Neural Networks (Univ. of Montreal)
Improvement of Network Models

<table>
<thead>
<tr>
<th>Model</th>
<th>AlexNet</th>
<th>VGG</th>
<th>GoogLeNet</th>
<th>ResNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization</td>
<td>Univ. of Tronto</td>
<td>Oxford Univ.</td>
<td>Google</td>
<td>Microsoft Research Asia</td>
</tr>
<tr>
<td>Year</td>
<td>2012</td>
<td>2014</td>
<td>2014</td>
<td>2015</td>
</tr>
<tr>
<td>ILSVRC* Error Rate</td>
<td>15.30%</td>
<td>7.33%</td>
<td>6.66%</td>
<td>3.57%</td>
</tr>
<tr>
<td># of Layers</td>
<td>8</td>
<td>19</td>
<td>22</td>
<td>152</td>
</tr>
<tr>
<td># of Parameters[M]</td>
<td>62.4</td>
<td>144</td>
<td>7.0</td>
<td>56.0</td>
</tr>
<tr>
<td># of Operations[B]</td>
<td>1.14</td>
<td>19.6</td>
<td>1.5</td>
<td>11.3</td>
</tr>
</tbody>
</table>

*) ILSVRC: ImageNet Large Scale Visual Recognition Challenge

Based on slides of Dr. Momose, Hokkaido Univ.
https://www.semiconportal.com/archive/contribution/applications/160804-neurochip2-2.html
## Road Scene Semantic Segmentation

<table>
<thead>
<tr>
<th>Network</th>
<th># of Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECONVNET</td>
<td>251M</td>
</tr>
<tr>
<td>FCN</td>
<td>134M</td>
</tr>
<tr>
<td>Ensemble</td>
<td>269M</td>
</tr>
<tr>
<td>T-Net*</td>
<td>1.4M</td>
</tr>
<tr>
<td>SegNet</td>
<td>30M</td>
</tr>
</tbody>
</table>

Deep Compression (Stanford)


Binarized Neural Networks (Univ. of Montreal)

- Neural networks with binary weights and activations (+1/-1) except for the first and the last layers

\[ x_0, x_1, x_2, \ldots, x_N \]

\[ u = w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_N x_N + 1 \text{ or } -1 \]

\[ s = \text{Sign}(u) \]

\[ +1 \text{ or } -1 \]

Error Rate

<table>
<thead>
<tr>
<th>Error Rate</th>
<th>MNIST (Handwritten digits)</th>
<th>SVHN (Street View House Numbers)</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNN (Theano)</td>
<td>0.96%</td>
<td>2.80%</td>
<td>11.40%</td>
</tr>
<tr>
<td>Maxout</td>
<td>0.94%</td>
<td>2.47%</td>
<td>11.68%</td>
</tr>
<tr>
<td>Gated pooling</td>
<td>-</td>
<td>1.69%</td>
<td>7.62%</td>
</tr>
</tbody>
</table>

Outline

• Technology Trends

• LOGNET: energy-efficient neural networks using logarithmic computation (Stanford Univ. & Toshiba) [ICASSP 2017] [arXiv:1603.01025]

• TDNN: Time-Domain Neural Network (Toshiba) [A-SSCC 2016]
Motivation of LOGNET

- To realize energy-efficient neural networks
  - Data representation with fewer bits
  - Eliminate multiplications

Proposal
- Use logarithmic encoding

Efficient implementation for dedicated HW in SoC rather than SW on GPU

<Conventional CNN>
- Weight: Float or Fixed
- $X \rightarrow X' \rightarrow \sum W \times x$

<CNN using logarithmic>
- Weight: log2 (Fixed)
- $X \rightarrow X' \rightarrow \sum W \ll x \rightarrow \text{Bit shift-Accumulate}$
Evaluation of Proposed 1

- From memory with SMALL bandwidth
  - \( \log_2 x \) 3 or 4b fixed
  - \( \sum W \ll x \) Bit shift-Accumulate
  - ReLU
  - Position of leftmost ‘1’
  - \( \log_2 x \) 3 or 4b fixed
  - To memory with SMALL bandwidth

**Evaluation results**: ILSVRC-2012 using Chainer

Top5 accuracy vs Full scale range: AlexNet
Top5 accuracy vs Full scale range: VGG16
Evaluation of Proposed 2

Log data representation for both weights and activations

- Top-5 accuracies after linear and log2 encoding on all layers’ weight without retraining

<table>
<thead>
<tr>
<th>Model</th>
<th>Float 32b</th>
<th>Lin. 4b</th>
<th>log2 4b</th>
<th>Lin. 5b</th>
<th>log2 5b</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>78.3%</td>
<td>1.6%</td>
<td>73.4%</td>
<td>71.0%</td>
<td>74.6%</td>
</tr>
<tr>
<td>VGG16</td>
<td>89.8%</td>
<td>0.5%</td>
<td>85.2%</td>
<td>83.2%</td>
<td>86.0%</td>
</tr>
</tbody>
</table>
Training with Logarithmic Representation

- **Training Algorithm**

Algorithm 1: Training a CNN with base-2 logarithmic representation. $C$ is the softmax loss for each minibatch. LogQuant$(x)$ quantizes $x$ in base-2 log-domain. The optimization step Update$(W_k, gW_k)$ updates the weights $W_k$ based on backpropagated gradients $gW_k$. We use the SGD with momentum and Adam rule.

**Require:** a minibatch of inputs and targets $(a_0, a^*)$, previous weights $W$.

**Ensure:** updated weights $W^{t+1}$

1. Computing the parameters’ gradient:
   1.1. Forward propagation:
   - for $k = 1$ to $L$ do
   - $W^q_k \leftarrow \text{LogQuant}(W_k)$
   - $a_k \leftarrow \text{ReLU}(a^q_{k-1}W^q_k)$
   - $a^q_k \leftarrow \text{LogQuant}(a_k)$
   - end for
   1.2. Backward propagation:
   - Compute $g_{aL} = \frac{\partial C}{\partial a_L}$ knowing $aL$ and $a^*$
   - for $k = L$ to 1 do
     - $g^q_{ak} \leftarrow \text{LogQuant}(g_{ak})$
   - $g_{a_{k-1}} \leftarrow g^q_{a_k}W_k^q$
   - $gW_k \leftarrow g^q_{a_k}a^q_{k-1}$
   - end for
   2. Accumulating the parameters’ gradient:
   - for $k = 1$ to $L$ do
   - $W^{t+1}_k \leftarrow \text{Update}(W_k, gW_k)$
   - end for

- **CIFAR10 database**
- **VGG-like network**

Quantize gradients
Enables end-to-end training using logarithmic representation at 5b level
Outline

• Technology Trends

• LOGNET: energy-efficient neural networks using logarithmic computation (Stanford Univ. & Toshiba) [ICASSP 2017]

• TDNN: Time-Domain Neural Network (Toshiba) [A-SSCC 2016]
Why the brain is so energy efficient?

**Brain**

- Weight is built into each synapse. Don’t need to move weight at all.

**Power efficient!!**

**Conventional computer**

- Load weight from memory for EVERY calculation.

**Power hungry!!**
## Energy consumption

<table>
<thead>
<tr>
<th>Operation</th>
<th>Relative Cost (Energy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 bit int ADD</td>
<td>1</td>
</tr>
<tr>
<td>32 bit int MULT</td>
<td>31</td>
</tr>
<tr>
<td>32 Register File</td>
<td>10</td>
</tr>
<tr>
<td>32 bit SRAM</td>
<td>50</td>
</tr>
<tr>
<td>32 bit DRAM Memory</td>
<td>6400</td>
</tr>
</tbody>
</table>

How about hardware efficiency?

e.g. The number of weights ➞ 100x

Need to have 100x processing elements. Because each processing element (PE) is dedicated to each weight.

Need to minimize each PE!!
Our strategy

• In order to maximize the energy efficiency, we propose to employ fully spatially unrolled architecture (like the brain).

• In order to minimize the hardware size, we propose to employ Time Domain Analog and digital Mixed Signal processing (TDAMS) [11].

[Diagram showing TDNN (Time Domain Neural Network)]

Value is represented by the time difference between edges.

Polarity: +, if P is earlier than N
- , if P is later than N
TDAMS — Convolution

$\begin{align*}
( W1 \times X_1' + W2 ) \times X_2' &= W1X1'X2' + W2X2' \\
X_{i}' &= \text{XOR}(X_{i}, X_{i+1}) \\
\text{SIGN}(W1X1 + W2X2)
\end{align*}$
TDAMS - ADD

+ 1

Delay time is proportional to value of resistance

- 1

T0 + 1

T0 - 1
TDAMS - Multiplication

\[ x = +1 \]

\[ x' = +1 \]
(Digital High)

\[ x = -1 \]

\[ x' = -1 \]
(Digital Low)

\[ T0 \times +1 \]

\[ T0 \times -1 \]
TDAMS ~ Activation (SIGN)

Positive
(+1)

Negative
(-1)

VIP  
VIN

positive

VIP  
VIN

negative
TDAMS — Convolution

\[ (W_1 \times X_1' + W_2) \times X_2' = W_1X_1'X_2' + W_2X_2' \]

\[ X_i' = \text{XOR}(X_i, X_{i+1}) \]

\[ \text{SIGN}(W_1X_1 + W_2X_2) \]
Chip photograph

65nm CMOS technology
# of processing elements: 32768
Experimental results

<table>
<thead>
<tr>
<th>Layer</th>
<th>Fully connected layer</th>
<th>Convolutional layer #1</th>
<th>Convolutional layer #2</th>
<th>Convolutional layer #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4th</td>
<td></td>
<td>128</td>
<td>3×3</td>
<td>64×3×3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128×3×3</td>
<td>64×5×5</td>
<td>64×11×11</td>
</tr>
<tr>
<td>3rd</td>
<td>Pooling layer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pooling layer #2</td>
<td>16×13×13</td>
<td>16×3×3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pooling layer #1</td>
<td>16×30×30</td>
<td>1×3×3</td>
<td></td>
</tr>
</tbody>
</table>

Image (Data) | Kernel (Weight)

![Image (Data) with Kernel (Weight)](image-url)
## Experimental results

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Description</th>
<th>Image (Data)</th>
<th>Kernel (Weight)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully connected layer</td>
<td></td>
<td>10</td>
<td>128</td>
</tr>
<tr>
<td>Pooling layer</td>
<td></td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>Convolutional layer #1</td>
<td></td>
<td>128</td>
<td>3x3</td>
</tr>
<tr>
<td>Convolutional layer #2</td>
<td></td>
<td>64x11x11</td>
<td>16x3x3</td>
</tr>
<tr>
<td>Pooling layer #1</td>
<td></td>
<td>16x13x13</td>
<td>3x3</td>
</tr>
<tr>
<td>Convolutional layer #1</td>
<td></td>
<td>16x30x30</td>
<td>1x3x3</td>
</tr>
</tbody>
</table>

*Executed on TDNN*
Experimental results

Measured

<table>
<thead>
<tr>
<th>Digit</th>
<th>Measured</th>
<th>Expected (Simulated on PC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>0</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td>4</td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
</tbody>
</table>

“7”: Correct!
“2”: Correct!
“0”: Correct!
“4”: Correct!

Tested 1000 images. Same accuracy as for simulation.
## Performance comparison

Energy efficiency is 10x better than ISSCC 2016[3].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tech.[nm]</strong></td>
<td>65</td>
<td>65</td>
<td>65</td>
<td>28</td>
<td>40</td>
</tr>
<tr>
<td><strong>Chip area [mm²]</strong></td>
<td>-</td>
<td>3.61&lt;sup&gt;a&lt;/sup&gt;</td>
<td>1.31&lt;sup&gt;a&lt;/sup&gt;</td>
<td>430</td>
<td>0.012</td>
</tr>
<tr>
<td><strong>Energy efficiency [TSOp/s/W]</strong></td>
<td>48.2&lt;sup&gt;b&lt;/sup&gt;</td>
<td>48.2&lt;sup&gt;b&lt;/sup&gt;</td>
<td>0.402</td>
<td>0.039&lt;sup&gt;[6]&lt;/sup&gt;</td>
<td>3.86&lt;sup&gt;c&lt;/sup&gt;</td>
</tr>
<tr>
<td><strong>Hardware efficiency&lt;sup&gt;d&lt;/sup&gt; [GE/PE]</strong></td>
<td>3</td>
<td>76.5</td>
<td>4641&lt;sup&gt;a&lt;/sup&gt;</td>
<td>6.5</td>
<td>288</td>
</tr>
</tbody>
</table>

<sup>a</sup> core area including SRAM,  <sup>b</sup> excludes external I/O,  <sup>c</sup> excludes CML  
<sup>d</sup> 1GE:1.44um² (65nm), 0.65 um² (40nm), 0.49 um² (28nm)

Blue print with ReRAM

TDNN

Conventional computer

Analog signal

1.5 $\mu m^2$ @28nm = 230M PEs / 4 cm$^2$

= 3 2-input NAND
+ memory cell (e.g. ReRAM)

$\gg$ 50 2-input NAND

1bit

1bit

~6bit

1bit

cf. ResNet* : 230M parameters

Summary

• **Efficient Implementations of DNN are required for embedded systems and edge devices**

• **Efficient Implementations**
  – Improvement of Network Models.
    • Simple Network Models (e.g. GoogLeNet, ResNet)
  – Reduction of Parameters (# of data, bit width) and Compression
    • Deep Compression (Stanford): Pruning, Quantization, Huffman Coding
    • Binarized Neural Networks

• **Efficient Hardware Implementations**
  – LOGNET: energy-efficient neural networks using logarithmic computation
  – TDNN (Time Domain Neural Network)
    • Fully spatially unrolled architecture (like the brain).
    • Time Domain Analog and digital Mixed Signal processing (TDAMS)