Going Deeper with Embedded FPGA Platform for Convolutional Neural Network

Jiantao Qiu\textsuperscript{1}, Jie Wang\textsuperscript{1}, Song Yao\textsuperscript{1}, Kaiyuan Guo\textsuperscript{1}, Boxun Li\textsuperscript{1}, Erjin Zhou\textsuperscript{1}, Jincheng Yu\textsuperscript{1}, Tianqi Tang\textsuperscript{1}, Ningyi Xu\textsuperscript{2}, Sen Song\textsuperscript{3}, Yu Wang\textsuperscript{1}, Huazhong Yang\textsuperscript{1}

\textsuperscript{1}Department of Electronic Engineering, Tsinghua University
\textsuperscript{2}Hardware Computing Group, Microsoft Research Asia
\textsuperscript{3}School of Medicine, Tsinghua University

Group URL: \url{http://nicsefc.ee.tsinghua.edu.cn}
\{songyao, yu-wang\}@mail.tsinghua.edu.cn

2016/2/22
Contents

• Deep Learning and Convolutional Neural Network
• Motivation
• Related Work
• Our Work: Angel-Eye
  – Overall Flow
  – V1: Architecture and Implementation Details
  – V1: Performance Comparison
  – V2: Brief introduction
• Open Question: Computation Granularity
Deep Learning

- Deep Learning: The new tide in artificial intelligence
  - Inspired by neuroscience
  - A collection of simple trainable mathematical units, which collaborate to compute a complicated function.
  - Deep Neural Network (DNN)/Recurrent Neural Network (RNN)/Long-Short Term Memory (LSTM)/Convolutional Neural Network (CNN)
Convolutional Neural Network (CNN)

- CNN: State-of-the-art in visual recognition applications

<table>
<thead>
<tr>
<th>Year</th>
<th>Team</th>
<th>Top-5 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>NEC</td>
<td>71.8%</td>
</tr>
<tr>
<td>2011</td>
<td>XRCE</td>
<td>74.2%</td>
</tr>
<tr>
<td>2012</td>
<td>SuperVision</td>
<td>84.7%</td>
</tr>
<tr>
<td>2013</td>
<td>Clarifai</td>
<td>88.3%</td>
</tr>
<tr>
<td>2014</td>
<td>GoogLeNet</td>
<td>93.3%</td>
</tr>
<tr>
<td>2015</td>
<td>MSRA</td>
<td>96.4%</td>
</tr>
</tbody>
</table>

Top-5 accuracy of image classification in Image-Net Large-Scale Visual Recognition Challenge (ILSVRC)
Contents

• Deep Learning and Convolutional Neural Network
• Motivation
• Related Work
• Our Work: Angel-Eye
  – Overall Flow
  – V1: Architecture and Implementation Details
  – V1: Performance Comparison
  – V2: Brief introduction
• Open Question: Computation Granularity
CNN: Mainstream in Computer Vision

- CNN: State-of-the-art in visual recognition applications

Pedestrian Detection [NUS2015]

Vehicle and Lane Detection [Stanford2015]

Tracking [UIUC2015]

Object detection

Posture estimation

Face recognition
CNN: High Complexity

- Conv Layers: bounded by computations

- FC Layers: bounded by memory access
Motivation

Why customized hardware?
• High complexity versus Limited energy
• CPU and GPU are not efficient enough

How to accelerate CNN with FPGA?
• High Complexity under Limited Resource
  • CNN Model Compression
  • Highly efficient computing units
  • Using convolver for FC layers
• High Complexity under Limited Bandwidth
  • CNN model compression
  • Shorter representations
  • Reducing memory access

CNN acceleration is more than hardware
Complete compilation tool is expected
Contents

• Deep Learning and Convolutional Neural Network
• Motivation
• Related Work
• Our Work: Angel-Eye
  – Overall Flow
  – V1: Architecture and Implementation Details
  – V1: Performance Comparison
  – V2: Brief introduction
• Open Question: Computation Granularity
Related Work

- **(Baidu Slides)**
- **(CAS) T. Chen et al.** DianNao: A small-footprint high-throughput accelerator for ubiquitous machine learning, ASPLOS 2014
- **(CAS) Z. Du et al.** Shidiannao: Shifting vision processing closer to the sensor, ISCA 2015
- **(Stanford) S. Han et al.** EIE: Efficient Inference Engine on Compressed Deep Neural Network, arxiv
Related Work

• Memory System Optimization
  – DianNao Series

DianNao ‘14
Single-chip CNN/DNN Accelerator

DaDianNao ‘14
Multi-chip CNN/DNN Accelerator

ShiDianNao ‘15
Single-chip CNN Accelerator for Visual Recognition Algorithms

PuDianNao ‘15
An ML accelerator which accommodates seven representative ML techniques (CNN/DNN included).

*1 Diannao: A small-footprint high-throughput accelerator for ubiquitous machine learning, Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam. ASPLOS ’14
*4 Shidiannao: Shifting vision processing closer to the sensor, Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, ISCA ’15
Related Work

- Memory System Optimization
  - DianNao Series

**How to solve the memory problem?**

**Strategy 1: Tiling and Data Reuse**
Cut down memory traffic

**Strategy 2: Storage Buffer**
Dedicated buffer for data reuse

**Strategy 3: On-Chip Memory**
Using on-chip memory to store all parameters

**Problem:** Using on-chip memory to store parameters in each layer of the CNN model, hard to be used for state-of-the-art large CNN models
Related Work

• Computing Engine Optimization

All existing work considers partial of the entire flow, and thus are hard to fully utilize hardware and achieve optimal energy efficiency.


Contents

• Deep Learning and Convolutional Neural Network
• Motivation
• Related Work
• **Our Work: Angel-Eye**
  – Overall Flow
  – V1: Architecture and Implementation Details
  – V1: Performance Comparison
  – V2: Brief introduction
• Open Question: Computation Granularity
Overall Flow

- Overall Flow of Angel-Eye

```
CNN Model
   Model compression
Compressed Floating-Point Model
      Data/weight quantization
Compressed Fixed-Point Model
         Compilation
Instructions
```

“Goal: accelerate fast algorithms.”
Model Compression

- Model Compression
  - Reducing complexity while maintaining comparable accuracy

- Singular Value Decomposition (SVD)
  - No demand for specific computation unit
  - Moderate compression
  - Computation model
    \[ f_{out} = W f_{in} + b \]
    \[ W \approx U_d S_d V_d \]
  - Storage complexity
    \[ O(n_{in} n_{out}) \]
    \[ O(d n_{in} + d n_{out}) \]
    \[ d \ll n_{in}, n_{out} \]

<table>
<thead>
<tr>
<th>Network</th>
<th>FC6</th>
<th># of total weights</th>
<th># of operations</th>
<th>Top-5 accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-16</td>
<td>25088 \times 4096</td>
<td>138.36M</td>
<td>30.94G</td>
<td>88.00%</td>
</tr>
<tr>
<td>VGG-16-SVD</td>
<td>25088 \times 500 + 500 \times 4096</td>
<td>50.18M</td>
<td>30.76G</td>
<td>87.96%</td>
</tr>
</tbody>
</table>
Data Quantization

- Data Quantization
  - Uses shorter fixed-point numbers

Resources needed by a multiplier

- Dynamic-Precision Data Quantization
  - Dynamic for different layer

Proposed Flow

- Offline parameter quantization
- Online data quantization
- Parameter dynamic range analysis
- Feature maps
- Layer 1
- Layer N
- Fixed-point CNN model
- Float-point CNN model
- Parameter and data quantization configuration
## Data Quantization

- Dynamic-Precision Data Quantization Results (Simulation results)

<table>
<thead>
<tr>
<th>Network</th>
<th>VGG16</th>
<th>VGG16-SVD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Bits</strong></td>
<td>Single-float</td>
<td>16</td>
</tr>
<tr>
<td><strong>Weight Bits</strong></td>
<td>Single-float</td>
<td>16</td>
</tr>
<tr>
<td><strong>Data Precision</strong></td>
<td>N/A</td>
<td>$2^{-2}$</td>
</tr>
<tr>
<td><strong>Weight Precision</strong></td>
<td>N/A</td>
<td>$2^{-15}$</td>
</tr>
<tr>
<td><strong>Top-1 Accuracy</strong></td>
<td>68.1%</td>
<td>68.0%</td>
</tr>
<tr>
<td><strong>Top-5 Accuracy</strong></td>
<td>88.0%</td>
<td>87.9%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network</th>
<th>CaffeNet</th>
<th>VGG16-SVD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Bits</strong></td>
<td>Single-float</td>
<td>16</td>
</tr>
<tr>
<td><strong>Weight Bits</strong></td>
<td>Single-float</td>
<td>16</td>
</tr>
<tr>
<td><strong>Data Precision</strong></td>
<td>N/A</td>
<td>Dynamic</td>
</tr>
<tr>
<td><strong>Weight Precision</strong></td>
<td>N/A</td>
<td>Dynamic</td>
</tr>
<tr>
<td><strong>Top-1 Accuracy</strong></td>
<td>53.9%</td>
<td>53.9%</td>
</tr>
<tr>
<td><strong>Top-5 Accuracy</strong></td>
<td>77.7%</td>
<td>77.1%</td>
</tr>
</tbody>
</table>
Instruction Set

• Coarse-grained Instructions

## Instruction Set Table

<table>
<thead>
<tr>
<th>Index</th>
<th>Pool Bypass</th>
<th>NL Bypass</th>
<th>Zero Switch</th>
<th>Result Shift</th>
<th>Bias Shift</th>
<th>Write En</th>
<th>PE En</th>
<th>Phase Type</th>
<th>Pic Num</th>
<th>Tile Size</th>
<th>Layer Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>No</td>
<td>2</td>
<td>First</td>
<td>2</td>
<td>Tr</td>
<td>Conv</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
<td>Bias</td>
<td>X</td>
<td>BS</td>
<td>No</td>
<td>2</td>
<td>Calculate</td>
<td>2</td>
<td>Tr</td>
<td>Conv</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>No</td>
<td>Zero</td>
<td>X</td>
<td>X</td>
<td>PE</td>
<td>2</td>
<td>Calculate</td>
<td>2</td>
<td>Tr</td>
<td>Conv</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>RS</td>
<td>X</td>
<td>DDR</td>
<td>2</td>
<td>Last</td>
<td>2</td>
<td>Tr</td>
<td>Conv</td>
</tr>
</tbody>
</table>

• **Hardware handles fine-grained operations**
  • Inst 1: commands Input Buffer to load all the needed data
  • Inst 2: starts calculating the four tiled blocks in the output layer
  • Inst 3: Write En is set as “PE” to command Output Buffer send the intermediate results back to the Pes
  • Inst 4: Write EN is set as “DDR” to command the Output Buffer write results back to the external memory (last layer)
Architecture and Implementation Details

- **Overall Architecture**

- **Processing System**
  - Flexibility
  - CPU + DDR
  - Scheduling operations
  - Prepare data and instructions
  - Realize Softmax function

- **Programmable Logic**
  - Hardware acceleration
  - Computing Complex + On-chip Buffers + Controller + DMA
  - Few Complex PEs

- **Achieve three-level parallelism**
  - Inter-output: multiple PEs
  - Intra-output
  - Operator-level

- **16-bit dynamic-precision data quantization**
• Achieve intra-output parallelism by placing multiple Convolvers
• Convolver: optimized for 3x3 convolution operation
• Adder Tree: sum up results of one convolution operation
• NL: supports non-linear function (ReLU)
• Pool: supports max-pooling
• Bias Shift & Data Shift: support dynamic-precision fixed-point numbers
• Line-buffer design
  – Optimized for 3x3 Convolver
  – Supports operator-level parallelism
Architecture and Implementation Details

- **Tiling and Data Reuse Strategy**

- **Using Convolver for FC layers**
  - FC layers are bandwidth-bounded
  - Convolvers are enough to compute FC layers
  - Save resource to accelerate Conv layers

*Figure 9: Data arrangement in external memory: (a) Linear arrangement; (b) DMA-oriented arrangement.*
## Performance Comparison

- **Performance and Energy Efficiency Comparison**

<table>
<thead>
<tr>
<th></th>
<th>Chakaradhar 2010</th>
<th>Gokhale 2014</th>
<th>Zhang 2015</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Platform</strong></td>
<td>Virtex 5 SX240t</td>
<td>Zynq XC7Z045</td>
<td>Virtex7 VX485t</td>
<td>Zynq XC7Z045</td>
</tr>
<tr>
<td><strong>Clock (MHz)</strong></td>
<td>120</td>
<td>150</td>
<td>100</td>
<td>150</td>
</tr>
<tr>
<td><strong>Bandwidth (GB/s)</strong></td>
<td>-</td>
<td>4.2</td>
<td>12.8</td>
<td>4.2</td>
</tr>
<tr>
<td><strong>Quantization</strong></td>
<td>48-bit fixed</td>
<td>16-bit fixed</td>
<td>32-bit float</td>
<td>16-bit fixed</td>
</tr>
<tr>
<td><strong>Problem Complexity (GOP)</strong></td>
<td>0.52</td>
<td>0.552</td>
<td>1.33</td>
<td>30.76</td>
</tr>
<tr>
<td><strong>Performance (GOP/s)</strong></td>
<td>16</td>
<td>23.18</td>
<td>61.62</td>
<td><strong>136.97 (Overall)</strong> 187.89 (Conv)</td>
</tr>
<tr>
<td><strong>Power (W)</strong></td>
<td>14</td>
<td>8</td>
<td>18.61</td>
<td>9.63</td>
</tr>
<tr>
<td><strong>Power Efficiency (GOP/J)</strong></td>
<td>1.14</td>
<td>2.90</td>
<td>3.31</td>
<td><strong>14.22 (Overall)</strong> 19.50 (Conv)</td>
</tr>
</tbody>
</table>
Angel-Eye V2

- Similar overall architecture
- Fully parameterized design
  - Supports different data quantization settings
  - Supports different PE and Convolver number
- Supports different Conv kernel size
- User-friendly compiler
- Fine-grained instructions
  - Increase the flexibility of compiler
  - More optimization in compiler back-end

<table>
<thead>
<tr>
<th>Platform</th>
<th>Performance</th>
<th>Power</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angel-Eye V2 on Xilinx 7020</td>
<td>30GOP/S</td>
<td>~2W</td>
<td>~40 dollar</td>
</tr>
<tr>
<td>Nvidia Tegra K1</td>
<td>60-90GOP/S</td>
<td>~10-15W</td>
<td>199 dollar</td>
</tr>
</tbody>
</table>
Angel-Eye V2: Face Det + Alig

- Overall Flow

  ![Diagram showing the overall flow of Angel-Eye V2](image)

  - **Input**: RGB image
  - **Face Detection**: Haar-like feature on ARM
  - **Face Alignment**: 9-layer CNN in FPGA

  Convolutional Neural Network (CNN) architecture:
  - 16Conv3x3 -> 16Conv3x3 -> 32Conv3x3 -> 32Conv3x3 -> 48Conv3x3 -> 64Conv3x3 -> 64Conv3x3 -> 80Conv3x3 -> 128Conv3x3 -> FC10

- 8-bit dynamic-precision quantization without fine-tuning

<table>
<thead>
<tr>
<th></th>
<th>27.3794</th>
<th>32.0091</th>
<th>66.1796</th>
<th>27.1025</th>
<th>45.1339</th>
<th>33.6290</th>
<th>67.4658</th>
<th>62.2705</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fixed-point Network</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Original Network</strong></td>
<td>26.0281</td>
<td>31.5535</td>
<td>65.0193</td>
<td>25.9672</td>
<td>43.4501</td>
<td>33.8175</td>
<td>66.2999</td>
<td>61.1216</td>
</tr>
</tbody>
</table>
Contents

- Deep Learning and Convolutional Neural Network
- Motivation
- Related Work
- Our Work: Angel-Eye
  - Overall Flow
  - V1: Architecture and Implementation Details
  - V1: Performance Comparison
  - V2: Brief introduction
- Open Question: Computation Granularity
Open question: Computation Granularity

- Computer Engine Architecture Comparison
  - [KAIST ISSCC2016] and ours

Few complex compute elements
Open question: Computation Granularity

- Computer Engine Architecture Comparison

Few complex PEs VS. Many simple PEs

Guess: Neural networks are highly predictable and serial. Few complex PEs can better utilize these characters.
Conclusion

• Deep Learning: Mainstream in AI
• Motivation
• Related Work
• Our Work: Angel-Eye
  – Overall Flow
  – V1: Architecture and Implementation Details
  – V1: Performance Comparison
  – V2: Brief introduction
• Open Question: Computation Granularity
Thanks!
Q&A