Design Space Exploration of FPGA-Based Deep Convolutional Neural Networks

Philipp Gysel
ECE Department
University of California, Davis
Feature Extraction Approaches

– Hand crafted features such as HoG and SIFT
– Automated features extraction using Convolutional Neural Networks

CNN Based Feature Extraction

– Very effective in different vision tasks
– Very high computational complexity
Precision – Depth Tradeoff


- **AlexNet**
- **VGG A**
- **VGG B**
- **VGG D**
- **VGG E**
- **GoogleNet**

- Higher Depth
- Higher Precision
- Higher Execution Time
- Higher Power Consumption

Mobile devices have to offload the computation to a cloud.
An energy efficient and fast implementation of DCNNs is very beneficial for mobile devices. This can be achieved by hardware based acceleration of DCNNs.
AlexNet

Convolutional Layers. Over 90% of computation time.

Fully Connected Layers. They can extract local and global features.

[Krizhevsky 2012]
2D Convolution

- Center element of kernel is placed on each pixel of Input Feature Map (IFM)
- Convolution Result:
  \[(4 \times 0) + (0 \times 0) + (0 \times 0) + (0 \times 0) + (0 \times 1) + (0 \times 1) + (0 \times 0) + (0 \times 1) + (-2 \times 4) = -8\]
3D Convolution

- Each Output Feature Map (OFM) is the result of a 3D convolution of the Input Feature Map (IFM) with a Kernel stack.

- Example

\[ OFM_A = 3DConv(IFM, Kernel_A) \]

\[ \forall i \in \{0, 1, ..., 255\}: OFM_i = 3DConv(IFM, Kernel_i) \]
Parallelism Sources

• Inter Layer Parallelism
  – Compute different OFMs in parallel

• Inter Output Parallelism
  – Compute one OFM in parallel

• Inter Kernel Parallelism
  – Compute one convolution in parallel
Design Philosophy

• DCNNs
  – Computation bound
  – Communication bound

• Computation – Communication balance
  – Memory model
  – Computation model
Tiling in Convolutional Layers

From Previous Layer

Input Feature Maps

IFM’s tile size: \( T_n \)

Column Tile Size: \( T_c \)

Row Tile Size: \( T_r \)

To Next Layer

OFM’s tile size: \( T_m \)

Output Feature Maps

\( N \) \( M \)

\( R \) \( C \)

\( Ti \) \( Tj \) \( K \) Kernels
The Architecture Template

- **Intra Kernel Parallelism**
  - PCE: Parallel Convolution Engine
  - Here the number of parallel multiplications ($T_k$) is 4.

- **Inter Kernel Parallelism**
  - Convolve different IFMs

- **Inter Output Parallelism**
  - PCEs with different weights
Computation Model

- Number of cycles in tiled model:
  \[ \text{Cycles} = \text{Rounds} \times \text{Operations per round} \]
- \( \text{Rounds} = \left[ \frac{M}{T_m} \right] \times \left[ \frac{N}{T_n} \right] \times \frac{RC}{T_r T_c} \times \left[ \frac{K}{T_i} \right] \times \left[ \frac{K}{T_i} \right] \)
- \( \text{Ops per round} = (T_r T_c \times \left[ \frac{T_i T_j}{T_k} \right] + P) \)
Memory Model

- Computation to communication ratio:
  \[
  CTC = \frac{\text{Total Computation}}{\text{Total Communication}} = \frac{\alpha_{in} \times \beta_{in} + \alpha_{out} \times \beta_{out} + \alpha_{wght} \times \beta_{wght}}{2 \times M \times N \times R \times C \times K \times K}
  \]

- Weight’s buffer size
  \[
  \beta_{wght} = T_m \times T_n \times T_i \times T_j
  \]

- Number of loads and stores of weights
  \[
  \alpha_{wght} = \frac{M}{T_m} \times \frac{N}{T_n} \times \frac{R}{T_r} \times \frac{C}{T_c} \times \frac{K}{T_i} \times \frac{K}{T_j}
  \]
Design Space Exploration

- **Goal**
  - Maximize throughput and CTC
- **Constraints**
  - Memory bandwidth
  - On-chip memory
  - Area limit (computation)
- **Approach**
  - Explore the design space for different values of $T_m, T_n, T_r, T_c, T_i$ and $T_j$.
## Re-configurability Effects (1)

<table>
<thead>
<tr>
<th>Layer</th>
<th>Dynamic $T_m$, $T_n$ and $T_k$</th>
<th>Dynamic $T_m$ and $T_n$</th>
<th>Fixed $T_m$, $T_n$ and $T_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$T_m$</td>
<td>$T_n$</td>
<td>$T_k$</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>24</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Towards a static solution
Re-configurability Effects (2)

– Dynamic re-configurability has a minimal effect on the performance.
# Performance Comparison (1)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>Fixed</td>
<td>Fixed</td>
<td>Fixed</td>
<td>32bits float</td>
<td>32bits float</td>
</tr>
<tr>
<td>Frequency</td>
<td>150 MHz</td>
<td>125 MHz</td>
<td>200 MHz</td>
<td>100 MHz</td>
<td>100 MHz</td>
</tr>
<tr>
<td>FPGA Chip</td>
<td>VLX240T</td>
<td>SX240T</td>
<td>SX240T</td>
<td>VX485T</td>
<td>VX485T</td>
</tr>
<tr>
<td>Performance</td>
<td>17 GOPs</td>
<td>7.0 GOPs</td>
<td>16 GOPs</td>
<td>61.62 GFLOPs</td>
<td>84.2 GFLOPs</td>
</tr>
<tr>
<td>GOPs/Slice</td>
<td>4.5E-04</td>
<td>1.9E-04</td>
<td>4.3E-04</td>
<td>8.12E-04</td>
<td>11.09E-04</td>
</tr>
</tbody>
</table>
Performance Comparison (2)

Design Space Exploration of FPGA-Based Deep Convolutional Neural Networks
Conclusion

• Template architecture for convolution acceleration
• Analytically characterize performance and memory requirements
• Expand the design space to find best architecture
  – Parallel Convolution Engines
  – Tiling scheme expanded to kernel level
• Simulation shows speedup of 1.4X ... 1.9X over existing accelerators
Thank you!
References