Using FPGAs to Accelerate Neural Network Inference

1st FPL Workshop on Reconfigurable Computing for Deep Learning (RC4DL)
8. September 2017, Ghent, Belgium

Associate Professor Magnus Jahre
Department of Computer Science
Norwegian University of Science and Technology

Many of the contributions discussed in this presentation have been developed in close collaboration with Xilinx Research Labs.
Outline

Part 1: CNN acceleration on FPGAs should exploit customization

Part 2: CNN customization possibilities

- Quantization
- Accelerator architecture
- FPGA-friendly network transformations
- Memory system challenges

Part 3: Conclusion
Outline

Part 1: CNN acceleration on FPGAs should exploit customization

Part 2: CNN customization possibilities
- Quantization
- Accelerator architecture
- FPGA-friendly network transformations
- Memory system challenges

Part 3: Conclusion
Properties of Acceleration-friendly Workloads

Lots of Compute and Parallelism

High arithmetic intensity

Few or predictable branches

Regular memory accesses

Roofline Model of a Platform

Performance (Operations/second)

Arithmetic Intensity (Operations/Byte)

Compute-bound

Memory-bound

Application A

Application B

Convolutional Neural Networks (CNNs)

AlexNet

Heavy Computation

“Cat”

Krizhevsky et al., “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS, 2012
A convolutional layer convolves a set of filters over the input data.

Reorganize data to create a matrix-matrix multiplication or a number of matrix-vector multiplications.

Potential problem: If the filters overlap, the input data is duplicated (in theory).
## Choice of Matrix Multiplication Algorithm

<table>
<thead>
<tr>
<th>Feature</th>
<th>Dense Matrix Multiplication</th>
<th>Sparse Matrix Multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lots of Compute and Parallelism</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>High arithmetic intensity</td>
<td>✓ (can have)</td>
<td>×</td>
</tr>
<tr>
<td>Few or predictable branches</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Regular memory accesses</td>
<td>✓</td>
<td>×</td>
</tr>
</tbody>
</table>

Choice of algorithm is a trade-off that depends on platform and input characteristics
# CNN Inference Platform Alternatives

<table>
<thead>
<tr>
<th>Metric</th>
<th>CPU</th>
<th>GPU</th>
<th>ASIC</th>
<th>FPGA</th>
<th>“Winner”</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Performance</td>
<td>0</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>GPU</td>
</tr>
<tr>
<td>Sufficient Performance</td>
<td>+</td>
<td>+</td>
<td>?</td>
<td>++</td>
<td>FPGA</td>
</tr>
<tr>
<td>High Energy Efficiency</td>
<td>0</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>ASIC</td>
</tr>
<tr>
<td>Short Development Time</td>
<td>++</td>
<td>++</td>
<td>?</td>
<td>+</td>
<td>CPU/GPU</td>
</tr>
<tr>
<td>Low Cost</td>
<td>++</td>
<td>+</td>
<td>?</td>
<td>+</td>
<td>CPU</td>
</tr>
</tbody>
</table>

Overall: No platform stands out as a clear winner across all metrics.

Differentiator: FPGAs can customize the architecture to fit the current problem.
CNN Customization Potential

Exploit the characteristics of neural networks

- Many different CNNs can provide similar accuracy
- Choose one that matches the strengths of the target platform
- Retraining may be necessary
- Potential synergies between CNN algorithm development and acceleration potential

Dimensions of customization

- Accelerator architecture (generic vs network-specific)
- Data type (fixed point vs floating point)
- Data precision (binary, 8 bit, 64 bit, etc.)
- FPGA-friendly network transformations
Outline

Part 1: CNN acceleration on FPGAs should exploit customization

Part 2: CNN customization possibilities

Quantization

Accelerator architecture

FPGA-friendly network transformations

Memory system challenges

Part 3: Conclusion
Redundancy and Quantization

Evidence of redundancy in trained networks
- sparsification, low-rank approximations, fault tolerance...

Reduced precision (quantization)
- Restrict weights and/or activations to Q-bit values
- HW benefits: Low-bitwidth datapaths, regular compute

Sung et al: Quantization works well when:
- ...the network is “big enough”
- ...the network is aware of quantization during training

“(...) the performance gap between the floating-point and the retrain-based ternary (+1, 0, -1) weight neural networks (...) almost vanishes in fully complex networks (...)”
(Sung et al, Resiliency of Deep NNs Under Quantization)
Binarized Neural Networks (BNNs)

The extreme case of quantization

- Permit only two values: +1 and -1
- Binary weights, binary activations

By Courbariaux and Hubara et al. (NIPS 2016)

- Open source training flow, based on Theano and Lasagne
- Layers: convolutional, fully-connected, batchnorm and maxpool

Close to state-of-the-art accuracy on smaller image classification benchmarks

- And getting steadily better at the bigger benchmarks.

<table>
<thead>
<tr>
<th>Quantization</th>
<th>MNIST</th>
<th>SVHN</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary weights, Binary activations</td>
<td>0.96%</td>
<td>2.53%</td>
<td>10.15%</td>
</tr>
<tr>
<td>Binary weights, FP activations</td>
<td>1.29%</td>
<td>2.30%</td>
<td>9.90%</td>
</tr>
<tr>
<td>FP weights, FP activations</td>
<td>0.94%</td>
<td>1.69%</td>
<td>7.62%</td>
</tr>
<tr>
<td>BNN accuracy loss</td>
<td>-0.2%</td>
<td>-0.84%</td>
<td>-2.53%</td>
</tr>
</tbody>
</table>

% classification error (lower is better)
BNN Performance Potential on FPGAs

fewer LUTs/op: higher peak performance

stay on-chip: achieve more of the peak

GOPS

10^5

10^3

10^1

0.125 1 8

Ops:Byte

16-bit ops

8-bit ops

1-bit ops

66 TOPS

1 TOPS

0.1 TOPS 40 TOPS
Outline

Part 1: CNN acceleration on FPGAs should exploit customization

Part 2: CNN customization possibilities
- Quantization
- Accelerator architecture
- FPGA-friendly network transformations
- Memory system challenges

Part 3: Conclusion
FINN’s Heterogeneous Streaming Architecture

One hardware layer per BNN layer

Heterogeneous: Avoid “one-size-fits-all” penalties (computation not uniform across layers)

Streaming: Maximize throughput and minimize latency (overlap communication and computation)

## Outline

<table>
<thead>
<tr>
<th>Part 1:</th>
<th>CNN acceleration on FPGAs should exploit customization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 2:</td>
<td>CNN customization possibilities</td>
</tr>
<tr>
<td></td>
<td>Quantization</td>
</tr>
<tr>
<td></td>
<td>Accelerator architecture</td>
</tr>
<tr>
<td></td>
<td><strong>FPGA-friendly network transformations</strong></td>
</tr>
<tr>
<td></td>
<td>Memory system challenges</td>
</tr>
<tr>
<td>Part 3:</td>
<td>Conclusion</td>
</tr>
</tbody>
</table>
Transformation Example: Streamlining

State-of-the-art quantized neural network methods still employ some floating point computations to improve accuracy:

- Batch normalization
- Alpha-scaling
- Non-integer quantization levels

Streamlining avoids floating point computations by:

- Viewing quantization as successive thresholding
- Moving and collapsing linear transformations
- Absorbing linear operations into thresholds

Fewer layers reduce FPGA resource consumption in network-specific architectures
Outline

Part 1: CNN acceleration on FPGAs should exploit customization

Part 2: CNN customization possibilities

- Quantization
- Accelerator architecture
- FPGA-friendly network transformations
- Memory system challenges

Part 3: Conclusion
Off-Chip Weight Storage

Large neural networks force weights to be stored off-chip

- Increases bandwidth needs
- Need to exploit Memory Level Parallelism (MLP) to maximize bandwidth utilization

Neural network structures can be made sparse

- Potential for reducing compute and memory requirements
- Efficiently exploiting sparsity is tricky

\[
\begin{bmatrix}
1.1 & 0 & 0 \\
0 & 2.2 & 3.3 \\
4.4 & 0 & 5.5
\end{bmatrix}
\]

\text{colptr} = \{0, 2, 3, 5\}
\text{values} = \{1.1, 4.4, 2.2, 3.3, 5.5\}
\text{rowind} = \{0, 2, 1, 1, 2\}

\text{for} (j=0 \text{ to } n-1)
\text{for} (i=\text{colptr}[j] \text{ to } \text{colptr}[j+1])
\quad y[\text{rowind}[i]] += \text{values}[i] \times x[j]
Accelerator Performance Tricks

Most of the random access vector is unused at any given time

- Use light-weight preprocessing to determine needed cache capacity
- Use spare on chip memory for something else

Overlap memory accesses and computation

- Balanced system: Accelerator compute capability should match memory subsystem performance
- Parallelism effectively hides the compute latency
- Exploit Memory Level Parallelism (MLP) to further improve memory bus utilization and performance
- Solution: Non-blocking caches


Outline

Part 1: CNN acceleration on FPGAs should exploit customization

Part 2: CNN customization possibilities
- Quantization
- Accelerator architecture
- FPGA-friendly network transformations
- Memory system challenges

Part 3: Conclusion
Conclusion

Deep learning is well suited for acceleration

- No compute platform is a clear winner across all performance metrics
- FPGAs excel when we can leverage heavily customized accelerators
- Need to identify neural networks with computation and memory patterns that are suitable to FPGA platform characteristics

Possibilities for customization

- Examples: Data type, precision, architecture and network transformations
- Significant potential for co-design of neural network algorithms and FPGA accelerators
The Telenor-NTNU AI-Lab

- To enable both **basic** and **applied research**
- To allow wide variety of research areas
- Maintain research at **highest international level**
- To enable **cross-disciplinary collaboration**
Thank You!

The following people have contributed to the ideas covered in this presentation:

• Yaman Umuroglu, NTNU
• Michaela Blott, Xilinx Research Labs
• Researchers at the NTNU Computer Architecture Lab
• Researchers at Xilinx Research Labs