Tool-flows for mapping CNNs into FPGAs: Trends and Challenges

Christos Bouganis
ccb98@ic.ac.uk

Intelligent Digital Systems Lab
Electrical and Electronic Engineering Department
Imperial College London
Artificial Intelligence - Machine Learning – Deep Neural Networks

Artificial Intelligence

Machine Learning

Deep Neural Networks

CNNs

YOLO v2
Convolutional Neural Network

INPUT 32x32

Feature Extractor

Fully connected
Convolutional Neural Network - Trends

ImageNet Challenge

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.
A Deep Learning Software Ecosystem

First Wave
- Caffe (UC Berkeley)
- torch (NYU / Facebook)
- theano (Univ. of Montreal)

Training and Inference

Second Wave
- Caffe2 (Facebook)
- PyTorch (Facebook)
- TensorFlow (Google)
- CoreML (Apple)

Neural Network Toolbox
- CNTK (Microsoft)
- TensorRT (NVIDIA)
- mxnet (Amazon)

CoreML (Apple)
CNN Deployment Flow

User Input

CNN Structure

Trained Weights

Deep Learning Framework

Optimised mapping

- CPU
- GPU
- TK1
- TX1 & TX2
- Qualcomm Snapdragon
- Apple A11
CNN Deployment Flow

User Input → CNN Structure → Trained Weights → Deep Learning Framework → Optimised mapping → FPGA platform resources → CPU → GPU → TK1 → TX1 & TX2 → Qualcomm Snapdragon → Apple A11 → FPGA
In the last few years, significant progress in generic FPGA HLS tools
  • Vivado HLS, Intel OpenCL, MaxCompiler, LegUp, etc.
  • Generate designs based on the mapping and scheduling of low-level primitive operations → Large design space.
  • Low-level entry point

CNN workloads are highly structured
  • Layers with pre-defined types and parametrisation

Opportunity for domain-specific frameworks for CNNs
  • Generate optimised architectures
## Existing CNN-to-FPGA tool-flows

<table>
<thead>
<tr>
<th>Name</th>
<th>Input description</th>
<th>Year</th>
<th>Publication</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpgaConvNet</td>
<td>Caffe &amp; Torch</td>
<td>May 2016</td>
<td>FCCM 2016</td>
</tr>
<tr>
<td>DeepBurning</td>
<td>Caffe</td>
<td>June 2016</td>
<td>DAC 2016</td>
</tr>
<tr>
<td>Angel-Eye</td>
<td>Proprietary</td>
<td>July 2016</td>
<td>FPGA 2016</td>
</tr>
<tr>
<td>ALAMO</td>
<td>Proprietary</td>
<td>August 2016</td>
<td>FPL 2016</td>
</tr>
<tr>
<td>DnnWeaver</td>
<td>Caffe</td>
<td>October 2016</td>
<td>MICRO 2016</td>
</tr>
<tr>
<td>Caffeine</td>
<td>Caffe</td>
<td>November 2016</td>
<td>ICCAD 2016</td>
</tr>
<tr>
<td>FINN</td>
<td>Theano</td>
<td>February 2017</td>
<td>FPGA 2017</td>
</tr>
<tr>
<td>FP-DNN</td>
<td>TensorFlow</td>
<td>May 2017</td>
<td>FCCM 2017</td>
</tr>
<tr>
<td>SysArrayAccel</td>
<td>C</td>
<td>June 2017</td>
<td>DAC 2017</td>
</tr>
</tbody>
</table>
Key aspects for consideration – How do they compare?

• **Key aspects for consideration**
  • Neural Network Support
  • Front-End Support
  • Design Portability
  • Hardware Architectures
  • Design Space Exploration
  • Precision support
  • Objective function
Mainstream models include
- CNNs
- Recurrent Neural Networks (RNNs)
- Binarised Neural Networks (BNNs)
- Ternary Neural Networks

DeepBurning and FP-DNN
- RNNs
- LSTMs
- Residual connections in CNNs (FP-DNN)
• Critical for reaching a wide user base
• Two options
  • Integration with existing frameworks
    • FpgaConvNet, DeepBurning, DNNWeaver Caffeine
  • Proprietary front-ends
    • SysArrayAccel, Angle-Eye, CNN RTL

Supported Front-Ends
**Design Portability**

**Def.:** The degree to which a tool-flow can target:

1. devices by multiple vendors and families
2. different setups (SoCs, host-FPGA servers, stand-alone FPGAs).

### Supported FPGAs vendors

- Supported toolflows (%): Xilinx > Intel

### Supported setup

- Supported toolflows (%): SoC > Standalone > Host + FPGA

*Highest portability: DnnWeaver*
Two types of architectures

1. *Streaming architectures*

2. *Single computation engine*
Hardware Architecture – Streaming Architectures

Characteristics:
- Coarse pipeline of hardware stages
- One hardware stage per layer
- fpgaConvNet, DeepBurning, FINN

Advantages:
+ Customisation
+ Concurrent execution of layers
+ Flexible allocation of resources per layer, tailored to the target network

Disadvantages:
- Flexibility
- New bitsream for each CNN → long compilation times
Hardware Architecture – Single Computation Engine

**Characteristics:**
- Processing element-based design
- Fixed architecture, time-shared between layers
- Control via microinstructions
- Angel-Eye, ALAMO, DnnWeaver, Caffeine, FP-DNN, SysArrayAccel

**Advantages:**
+ Flexibility
+ One bistream can target many CNNs

**Disadvantages:**
- Customisation
- High performance variability across CNNs
- Inefficiencies due to processor-like control mechanisms
Each toolflow defines an architectural design space

- **Parameter Space:**
  - Which trade-offs and alternative designs can be explored?

- **Exploration Method**
  - How it the design space traversed?
  - Which objectives are optimised?

**Observations**

- Toolflows with streaming architectures define a finer-grained space.
  - Structure of the pipeline
  - Allocation of resources among hardware stages
- Single computation engine toolflows focus on:
  - Scaling of the computation engine with HW resources
  - CNN-to-microinstructions mapping
## Design Space Exploration

<table>
<thead>
<tr>
<th>Toolflow</th>
<th>Formulation</th>
<th>Solver</th>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpgaConvNet</td>
<td>Mathematical optimisation s.t. the rsc budget</td>
<td>Global Optimiser (Simulated Annealing)</td>
<td>Throughput, latency or multiobjective criteria</td>
</tr>
<tr>
<td>DnnWeaver</td>
<td>-/-</td>
<td>Heuristic</td>
<td>Throughput</td>
</tr>
<tr>
<td>Caffeine</td>
<td>Roofline model</td>
<td>Enumeration</td>
<td>Throughput</td>
</tr>
<tr>
<td>SysArrayAccel</td>
<td>Mathematical optimisation s.t. the rsc budget</td>
<td>Pruning + Enumeration</td>
<td>Throughput</td>
</tr>
<tr>
<td>FINN</td>
<td>Rule-based: rate-balancing</td>
<td>Heuristic</td>
<td>Throughput, latency</td>
</tr>
<tr>
<td>FP-DNN</td>
<td>Rule-based: bandwidth-matching</td>
<td>Heuristic</td>
<td>Throughput</td>
</tr>
<tr>
<td>ALAMO</td>
<td></td>
<td>Heuristic</td>
<td>Throughput, latency</td>
</tr>
<tr>
<td>DeepBurning</td>
<td>Rule-based: rate-balancing</td>
<td>Heuristic</td>
<td>Throughput, latency</td>
</tr>
<tr>
<td>Angel-Eye</td>
<td>Rule-based</td>
<td>Heuristic</td>
<td>Throughput, latency</td>
</tr>
</tbody>
</table>
Arithmetic Precision

Deep learning community

- Full precision: FP32
- Binary/Ternary

Automated

User-specified

Uniform precision
- fpgaConvNet
- FP-DNN
- SysArrayAccel

Variable precision
- ALAMO
- DnnWeaver
- DeepBurning

Dynamic fixed-point
- Angel-Eye

Binary
- FINN
Achieved QoR is the most critical factor.

QoR can be evaluated with respect to two factors:
1) Comparison of QoR between toolflows for the same CNN-FPGA pair,
2) Comparison with hand-tuned accelerator for the same CNN-FPGA pair.

Fair comparisons: Each toolflow to target the same CNN-FPGA pair.
However, the majority of existing toolflows are not open-sourced, or provide limited support.

- DnnWeaver targeting Zynq XC7Z020 (limited support, open sourced)
- FINN targeting Xilinx PYNQ-Z1 board (specific BNNs)
- Angel-Eye used internally by DeePhi.
- fpgaConvNet (webpage with up-to-date benchmarking results)
High-level Performance Observations

1) RTL-based designs tend to outperform their HLS counterparts.
2) Finer-grained DSE tends to offer an advantage in terms of obtained performance.
3) Single computation engines tend to reach higher performance for CNNs with a uniform structure.
4) Comparable or even better performance than hand-tuned designs
THE FUTURE OF CNN-TO-FPGA TOOLFLOWS
**Objective 1.** Targeting next-generation CNN models.

**Trends:**

1) Increased depth.

2) Increased workload.
   - 20x in GOps/input from AlexNet (2012) to VGG-16 (2014)

3) Novel compound components.
   - Inception module (GoogLeNet), residual block (ResNet), dense block (DenseNet), residual Inception block (Inception-v4).

**Challenge**

- Irregular layer connectivity challenges the automation of mapping to hardware.
Objective 1. Targeting next-generation CNN models.
Support for Recurrent Neural Networks (RNNs):
  • TPU paper by Google – around 95% of NN workloads are RNNs.
Challenge
  • RNNs are memory-bounded and require different design approach to CNNs.
**Objective 2. Support for compressed and sparse CNNs**

Numerous schemes exploit redundancy in the model to reduce inference time.

- Low-rank approximations, pruning, sparsification, quantisation.
- ASIC accelerators have introduced designs for such networks (e.g. zero-skipping compute units, bit-serial arithmetic, etc.)

**Challenge**

- Methods such as pruning can break the uniformity of computation and memory accesses.
Objectives of a CNN-to-FPGA Toolflow

Objective 3. FPGA-based CNN training

GPUs are the current standard for CNN training.

- Next-generation FPGAs demonstrate promising performance and power efficiency (Stratix 10, UltraScale).
- Recent advances in low-precision NN training offers space for customisation and variable-precision arithmetic that suits FPGAs.
- Slightly explored by the F-CNN framework.

Challenge

- Demonstrate gains of FPGAs over GPUs for CNN training.
Objectives of a CNN-to-FPGA Toolflow

Objective 4. Hardware-Network co-design
End-to-end toolchain, from dataset and application to network and hardware design.

- Expose hardware performance and power consumption to the training phase, co-optimise the CNN model and the hardware under a holistic view.

Challenge

- Long-term objective for the community towards the efficient hardware execution of high-performing neural networks.
In the past decade, Convolutional Neural Networks (CNNs) have demonstrated state-of-the-art performance in various Artificial Intelligence tasks. To accelerate the experimentation and development of CNNs, several software frameworks have been released, primarily targeting power-hungry CPUs and GPUs. In this context, reconfigurable hardware in the form of FPGAs constitutes a potential alternative platform that can be integrated in the existing CNN ecosystem to provide a tunable balance between performance, power consumption and programmability. In this paper, a survey of the existing CNN-to-FPGA toolflows is presented, comprising a comparative study of their key characteristics which include the supported applications, architectural choices, design space exploration methods and achieved performance. Moreover, major challenges and objectives introduced by the latest trends in CNN algorithmic research are identified and presented. Finally, an evaluation methodology is proposed, aiming at the comprehensive, complete and in-depth evaluation of CNN-to-FPGA toolflows.
References


- Ying Wang, Jie Xu, Yinhe Han, Huawei Li, and Xiaowei Li. 2016. DeepBurning: Automatic Generation of FPGA-based Learning Accelerators for the Neural Network Family. In DAC. Article 110. DOI:http://dx.doi.org/10.1145/2897937.2898003


- Yufei Ma, Naveen Suda, Yu Cao, Jae sun Seo, and Sarma Vrudhula. 2016. Scalable and Modularized RTL Compilation of Convolutional Neural Networks onto FPGA. In FPL. DOI:http://dx.doi.org/10.1109/FPL.2016.7577356

References

- Hardik Sharma, Jongse Park, Divya Mahajan, Emmanuel Amaro, Joon Kyung Kim, Chenkai Shao, Asit Mishra, and Hadi Esmaeilzadeh. 2016. From High-Level Deep Neural Models to FPGAs. In MICRO. DOI:http://dx.doi.org/10.1109/micro.2016.7783720


- Nicholas J. Fraser, Yaman Umuroglu, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. 2017. Scaling Binarized Neural Networks on Reconfigurable Logic. In PARMA-DITAM. DOI:http://dx.doi.org/10.1145/3029580.3029586
