The Evolution of Accelerators upon Deep Learning Algorithms

Song Yao, Shuang Liang, Junbin Wang, Zhongmin Chen, Shaoxia Fang, Lingzhi Sui, Qian Yu, Dongliang Xie, Xiaoming Sun, Song Han, Yi shan, and Yu Wang

August 21st, 2018, Cupertino

*The authors represent the combined efforts from many talented and hard-working engineers in DeePhi
01  What’s New About DeePhi Tech
02  The Evolution of Algorithms
03  The Evolution of DeePhi’s DPU
04  DL Software: New Changes
05  Summary
Part 1

What’s New About DeePhi Tech
DeePhi: Now Part of Xilinx
Long History, Close Collaboration, and Better Future

Collaboration with Xilinx University Program
Deep learning acceleration
Time series analysis
Stereo vision
……

Development of products on Xilinx FPGA platform since inception of DeePhi
Face recognition
Video analysis
Speech recognition acceleration
……

Co-Marketing and Co-Sales with Xilinx Team
Data Center
Automotive
Video surveillance
……

Deep learning acceleration
Time series analysis
Stereo vision
……

Face recognition
Video analysis
Speech recognition acceleration
……

Data Center
Automotive
Video surveillance
……
We Provide Full-Stack Deep Learning Solutions

Models
- Face detection
- Pose estimation
- Video analytics
- Lane detection
- Vehicle detection
- Segmentation

Framework
- Caffe
- TensorFlow
- mxnet

Tools
- DEEPHi
- DNNDK

IP & AI Platforms
- Z7020
- Z7020
- ZU2
- ZU2
- ZU9
DeePhi’s Solution now on AWS and Huawei Cloud!

DeePhi Descartes Efficient Speech Recognition Engine

Sold by: Beijing DeePhi Technology Co., Ltd. Latest Version: 2018.02.2b
DDESE is an efficient end-to-end automatic speech recognition (ASR) engine with the deep learning acceleration solution of algorithm, software and hardware co-design (containing Linux/Unix 0)

Overview  Pricing  Usage  Support

Product Overview

This is an end-to-end ASR (Automatic Speech Recognition) system with FPGA acceleration on AWS F1 by DeePhi. We modify the Baidu DeepSpeech2 framework (https://github.com/SeanNaren/deepspeech2-pytorch) for our solution of algorithm, software and hardware co-design, using LibrirSpeech 1000th dataset (http://www.openslr.org/12/) for model training and compression. Our model consists of 2 convolution layers (with Batch Normalization and Hardtanh), 5 bi-directional LSTM layers and 1 fully connected layer, together with a Softmax layer. We mainly focus on the acceleration of CNN and LSTM layers by FPGA, while other parts are implemented on CPU. For a test audio of 1 second, we are able to achieve a latency of 20.59ms for the entire end-to-end ASR system on AWS F1 with the help of our acceleration, which is about 2.0X speedup compared to cudnn solution tested locally on GPU P4. Users could run the test scripts for both performance comparisons of CPU/FPGA and single sentence recognition.

Highlights

- Support CNN and bi-directional LSTM for model inference.
- Support testing for both performance on CPU/FPGA and single sentence recognition.
- Support your own test audio rate at 16kHz sample rate, no longer than 16 seconds.

https://aws.amazon.com/marketplace/pp/B079N2J42R?from=timeline&isappinstalled=0
https://market.huaweicloud.com/product/00301-110982-0--0
Part 2

Evolution of Algorithms
In 2018, We Are Still Using Old Algorithms as Benchmark

LeNet-5: 1998

AlexNet: 2012

VGG-Net: 2014

GoogLeNet: 2014

ResNet: 2015
Changes in 2015-2018 Are More than That before 2015

ReLU is not the only widely used activation function

Group Conv: Not every input channel is connected to every output channel

Channels can have different weights

Depth-wise Conv and point-wise Conv are widely used
Changes in 2015-2018 Are More than That before 2015

DenseNet: Layers are not necessarily serial

Dilated Conv: Conv kernels are not necessarily dense

Deformable Conv: Conv kernels are not necessarily rectangle
Part 3

Evolution of DeePhi’s DPU
## From DPU-V1 to DPU-V2.5

<table>
<thead>
<tr>
<th>Architecture (single core)</th>
<th>DPU-V1</th>
<th>DPU-V2</th>
<th>DPU-V2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak performance (GOPS)</td>
<td>120</td>
<td>1350</td>
<td>1367</td>
</tr>
<tr>
<td>On-chip Memory (KB)</td>
<td>300</td>
<td>1123</td>
<td>1123</td>
</tr>
<tr>
<td>LUT</td>
<td>30k</td>
<td>75k</td>
<td>37k</td>
</tr>
<tr>
<td>FF</td>
<td>35k</td>
<td>146k</td>
<td>80k</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>214</td>
<td>333</td>
<td>333</td>
</tr>
<tr>
<td>Typical FPGA Platform</td>
<td>Zynq 7020 (28nm)</td>
<td>ZU9 (16nm)</td>
<td>ZU9 (16nm)</td>
</tr>
<tr>
<td>Total Cores</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Total Peak Perf (GOPS)</td>
<td>120</td>
<td>2700</td>
<td>4100</td>
</tr>
</tbody>
</table>

**Direction of improvements:**

Better scheduling strategy, higher resource utilization, more supported features, and more flexibility
From DPU-V1 to DPU-V2.5

DPU-V1

1. Convolution
   (a) Kernel size=3*3
   (b) Stride=1
   (c) Arbitrary padding size

2. Pooling
   (a) Kernel size=2*2
   (b) Stride=2

3. Activation function
   (a) ReLU

Supported network example: VGG16

DPU-V2

1. Feature map
   (a) Elementwise

2. Convolution
   (a) Arbitrary kernel size
   (b) Arbitrary stride
   (c) Arbitrary padding size

3. Pooling
   (a) avg/max pooling
   (b) Size=2*2,3*3
   (c) Stride=1,2
   (d) Arbitrary padding size

4. Activation function: (a) ReLU

5. FC(INT8)

6. Multi-Batch

Supported network example: LeNet/RosNet50/InceptionV1/InceptionV2/MobileNet/YOLO*SegNet*FPN* note: the networks with * is only supported on DPU-V2.5
From DPU-V1 to DPU-V2.5

1. Feature map
   (a) Elementwise
   (b) Split
   (c) Concat
   (d) Resize
   (e) Batch Normalization

2. Convolution
   (a) arbitrary kernel size
   (b) arbitrary stride
   (c) arbitrary padding size
   (d) Deconv
   (e) Dilated Conv
   (f) Depthwise Conv

3. Pooling
   (a) avg/max pooling
   (b) arbitrary size
   (c) arbitrary stride
   (d) arbitrary padding
   (e) ROI pooling

4. Activation function
   (a) ReLU
   (b) PReLU
   (c) LeakyReLU
   (d) Sigmoid

5. FC (INT8/FP32)

6. Multi-Batch
Case Study: Depth-Wise (DW) Convolution

(a) Classic Convolution

(b) Depth-wise Separable Convolution

Depth-Wise: Number of groups = Number of channels
Efficient Networks May Not Be Friendly to Hardware

Times of improvement of MobileNet-v1/v2 over ResNet-50 on Tesla P100 GPU

<table>
<thead>
<tr>
<th>Model</th>
<th># Parameters</th>
<th># Computations</th>
<th>Runtime (ms)</th>
<th>Reference Top-1 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>25.5M</td>
<td>7.72 GOPS</td>
<td>4.20</td>
<td>76.1%</td>
</tr>
<tr>
<td>MobileNet-v1</td>
<td>4.2M</td>
<td>1.14 GOPS</td>
<td>1.37</td>
<td>70.6%</td>
</tr>
<tr>
<td>MobileNet-v2</td>
<td>3.5M</td>
<td>0.60 GOPS</td>
<td>1.49</td>
<td>72.0%</td>
</tr>
</tbody>
</table>
Efficient Networks May Not Be Friendly to Hardware

Times of improvement of MobileNet-v1 over three benchmark networks on DeePhi’s DPU on ZU9 FPGA

<table>
<thead>
<tr>
<th>Model</th>
<th>Computations</th>
<th>Reference Top-1 Accuracy</th>
<th>Performance</th>
<th>FPS</th>
<th>Utilization Rate</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-16</td>
<td>30.7 GOPS</td>
<td>71.9%</td>
<td>2.36 TOPS</td>
<td>76.9</td>
<td>87.30%</td>
<td>2013</td>
</tr>
<tr>
<td>GoogleNet</td>
<td>3.89 GOPS</td>
<td>71.0%</td>
<td>0.99 TOPS</td>
<td>254</td>
<td>36.80%</td>
<td>2014</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>7.72 GOPS</td>
<td>76.1%</td>
<td>1.06 TOPS</td>
<td>137</td>
<td>39.30%</td>
<td>2015</td>
</tr>
<tr>
<td>MobileNet-v1</td>
<td>1.14 GOPS</td>
<td>70.6%</td>
<td>0.77 TOPS</td>
<td>675</td>
<td>28.62%</td>
<td>2017</td>
</tr>
</tbody>
</table>
Why Depth-Wise Convolution so Slow?

- The Communication/Computation (CTC) Ratio of depth-wise is very high
- Point-wise (1x1, PW) conv is memory-bound
Our Strategy Resolving the Problem

(a) Fused-layer Convolution for MobileNet

(b) The workload ratio between adjacent PW-Conv and DW-Conv

(c) Computing Architecture
Utilization Improvement on MobileNet-v1 with Layer Fusion

Results on ZU9 FPGA

- Dual cores on ZU9 FPGA
- Each core uses 1024KB BRAM
- In total 3648KB BRAM on ZU9 FPGA

Simulation results with doubled BRAM

- Each core uses 2048KB BRAM
Part 4

DL Software: New Changes
DNNDK: Make it Easy to Develop

DL Application

DL Framework

DPU Platform

DNNDK

Caffe

TensorFlow

mxnet

Compression
- Pruning
- Quantization

Compilation
- Compiler
- Assembler

Runtime
- Core API
- Loader
- Driver
- Tracer

Pruning
Quantization
Compiler
Assembler
Core API
Loader
Driver
Tracer

Compiler
Assembler
Core API
Loader
Driver
Tracer

© 2018 DeePhi Tech. All Rights Reserved.
Node fusion/decomposition & data stream optimization

More Optimizations Should be Considered in Compiler

Vertical fusion

Conv

Relu

Pool

Conv+Relu+Pool

Conv+Relu+Pool

Elementwise

Conv+Elementwise

Horizontal fusion

Conv

Conv

Conv+Conv

Decomposition

Conv

Conv

Conv

Conv

Concat

Pool

Conv+Pool

Conv+Pool

Memory allocation/scheduling/reuse to improve performance
Part 5
Summary

Discover the Philosophy behind Deep Learning Computing
Summary

• Algorithms are evolving at an increasingly faster rate
• Modern neural networks like MobileNet are not necessarily friendly for hardware acceleration on existing ASICs
• We propose a fusing-layer strategy together with compiling optimization to better accelerate Depth-wise/Point-wise convolutions
• More optimization strategies for new types of networks should be considered

• Pure hardware evolves slowly due to the long period in designing and manufacturing chips
• FPGA can be benefited from latest DL techniques in both hardware and software side
THANK YOU!

Friday, August 17, 2018

Discover the Philosophy behind Deep Learning Computing