Fast Monocular Depth Estimation on an FPGA

2020.5

Tokyo Institute of Technology, Dept. of Information and Communications
Youki Sada, Naoto Soga, Masayuki Shimoda, Akira Jinguji, Shimpei Sato, Hiroki Nakahara
sada@reconf.ict.e.titech.ac.jp, nakahara@ict.e.titech.ac.jp
Outline

1. Project background
   • Monocular depth estimation

2. CNN(Convolutional Neural Network) scheme

3. FPGA implementation
   • Architecture of convolutional circuit
   • Experimental results (accuracy / processing speed)
Monocular Depth Estimation

• Estimate the depth from a single RGB image

• Applications: Driving automation system, robotics

Source: arxiv, Ibraheem et. al, High Quality Monocular Depth Estimation via Transfer Learning
Monocular Depth Estimation (Contd.)

• **Depth sensors**
  
  - LiDAR
  
  - Stereo camera
  
  \[\rightarrow\] Expensive, lack of depth data

• **CNN scheme**
  
  - Dense Depth Map
  
  - **Use of a monocular RGB camera is valuable**

\[\rightarrow\] Implement monocular depth estimation on an **FPGA** for light-weight and low-cost
Outline

1. Project background
   - Monocular depth estimation

2. CNN(Convolutional Neural Network) scheme

3. FPGA implementation
   - Architecture of convolutional circuit
   - Experimental results (accuracy / processing speed)
**CNN model**

- **Convolutional layer only**
  - Depth-CNN (our CNN model)
    - MobileNetV1 base, A skip connection, Atrous spatial pyramid pooling (used by SOTA models)
  → Realistic accuracy with few parameters

---

Optimization: Weight Pruning & Quantization

- Parameters and OPs on PwConv is account for >90%
  
  → Apply weight pruning on PwConv (Hardware oriented filter-wise pruning)

- Weight: 8-bit, Activation: 6-bit Quantization for low-cost FPGA

---

PW; PwConvs, others; Conventional Convs, DwConvs, and Atrous DwConvs
Outline

1. Project background
   • Monocular depth estimation

2. CNN(Convolutional Neural Network) scheme

3. FPGA implementation
   • Architecture of convolutional circuit
   • Experimental results (accuracy / processing speed)
Overall Architecture

- Single computational engine scheme
  - general convolution, Depthwise, Atrous Depthwise, Pointwise, Pruned Pointwise

- All calculation results (Feature maps) are stored on on-chip memory
Outline

1. Project background
   • Monocular depth estimation

2. CNN(Convolutional Neural Network) scheme

3. FPGA implementation
   • Architecture of convolutional circuit
   • Experimental results (accuracy / processing speed)
Experiment

• **Dataset: NYU-Depth v2**
  - Generated by Microsoft Kinect
  - Indoor scene images (RGB + Depth Map)
  - 50,688 train images, 654 test images

• **FPGA system: Avnet Ultra96**
  - Xilinx Ultrascale+ MPSoC ZU3EG
  - [PYNQ Framework and VIVADO HLS](#)
  - Ubuntu

• **GPU system: NVIDIA Jetson TX2**
  - Tensor RT
  - Ubuntu
Experimental result

Accuracy comparison with conventional CNNs

<table>
<thead>
<tr>
<th></th>
<th>Resol.</th>
<th>weights</th>
<th>OP[G]</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigen[1] (VGG)</td>
<td>228x304</td>
<td>dense</td>
<td>23.4</td>
<td>76.9%</td>
</tr>
<tr>
<td>Laina[2] (ResNet50)</td>
<td>228x304</td>
<td>dense</td>
<td>22.9</td>
<td>78.9%</td>
</tr>
<tr>
<td><strong>Ours</strong></td>
<td>256x256</td>
<td>dense</td>
<td>0.6</td>
<td><strong>77.6%</strong></td>
</tr>
<tr>
<td><strong>Ours</strong> (wt. 8, act. 6, zero wt. 87%)</td>
<td>256x256</td>
<td>sparse</td>
<td>0.1</td>
<td>76.2%</td>
</tr>
</tbody>
</table>

Comparison for proposed compression

<table>
<thead>
<tr>
<th>Image size</th>
<th>Precision</th>
<th>Weights</th>
<th>Size[Mb]</th>
<th>OP[M]</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>256x256</td>
<td>FP32</td>
<td>dense</td>
<td>56.4</td>
<td>0.6</td>
<td>77.6%</td>
</tr>
<tr>
<td>256x256</td>
<td>FP32</td>
<td>Sparse</td>
<td>7.2</td>
<td>0.1</td>
<td>76.2%</td>
</tr>
<tr>
<td>256x256</td>
<td>wt. 8, act. 6</td>
<td>Sparse</td>
<td><strong>1.8</strong></td>
<td>0.1</td>
<td>76.2%</td>
</tr>
</tbody>
</table>

Achieved X6 fewer OPs and X31 smaller CNN weight size (only 1.4% accuracy degradation)

[1] D. Eigen, C. Puhrsch, and R. Fergus,

Experimental result (Contd.)

Utilization result by ZU3EG FPGA

<table>
<thead>
<tr>
<th>BRAM</th>
<th>DSP</th>
<th>LUT</th>
<th>FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>380 (90%)</td>
<td>198 (55%)</td>
<td>45,666 (65%)</td>
<td>23,254 (16%)</td>
</tr>
</tbody>
</table>

Comparison with GPU

<table>
<thead>
<tr>
<th>Platform</th>
<th>Compiler</th>
<th>Clock Freq.</th>
<th>Speed [avg. FPS]</th>
<th>Power [W]</th>
<th>Efficiency [FPS/W]</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA (Ultra96)</td>
<td>VIVADO HLS</td>
<td>0.2 GHz</td>
<td>123.6</td>
<td>0.3</td>
<td>412</td>
</tr>
<tr>
<td>GPU (Jetson TX2)</td>
<td>TensorRT</td>
<td>1.3 GHz</td>
<td>79.8</td>
<td>6.0</td>
<td>13</td>
</tr>
</tbody>
</table>

Compared to the GPU, the FPGA was **X1.5 faster** and **X31 better** performance per power.

*Our proposed monocular depth estimation is suitable for embedded systems*
Summary

• Propose lightweight CNN-based monocular depth estimation, which is suitable for embedded systems

• Achieved 6X smaller OPs by hardware-oriented weight pruning

• Implemented on Ultra96 FPGA board and compared with mobile GPU (Jetson TX2)
  • Our proposed FPGA system achieved higher efficiency, better speed, and lower power