Acceleration of Deep Learning Inference on Raspberry Pi’s VideoCore GPU

Koichi Nakamura
Idein Inc/Co-Founder, CEO
Koichi Nakamura
Idein/C-Founder, CEO

Majored in computer science at the Graduate School of Information Science and Technology, the University of Tokyo, and monstly studied optimizing compiler theory for massively parallel computers. In 2015, left the university and founded Idein. Selected as an ARM Innovator in 2018.
Idein Inc.

Mission
Enabling Software to Utilize Any Physical-World Data

Core Technology
Accelerating Deep-learning for Embedded device

Business
Edge AI Platform Actcast (actcast.io)

<table>
<thead>
<tr>
<th>Name</th>
<th>Idein Inc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Establishment</td>
<td>April 2015</td>
</tr>
<tr>
<td>Location</td>
<td>Tokyo</td>
</tr>
<tr>
<td></td>
<td>Jinbo-Cho Chiyoda-Ku</td>
</tr>
<tr>
<td># of Employees</td>
<td>48 (as of Nov 2020)</td>
</tr>
<tr>
<td>Total Fund Raising</td>
<td>$33M</td>
</tr>
</tbody>
</table>

Selected by J-Startup arm AI Partner
Demo Videos
ImageNet Classification
1000 classes
MobileNet V2 1.0
224x224, fp32

> 8fps on Raspberry Pi Zero
Object Detection
1 class, MobileNet V2 SSD, 196x196, fp32

> 12fps on Raspberry Pi 3
Pose Estimation
Pose Proposal Network, 224x224, fp32

> 9fps on Raspberry Pi 3
Semantic Segmentation
DeepLab v3+ with MobileNet v2 backbone, 256x256, fp32

> 5fps on Raspberry Pi 3
Image Captioning
Encoder: MobileNet V2
Decoder: LSTM
224x224, fp32

> 2.5fps on Raspberry Pi 3
Sound Recognition
521 classes

> 5fps on Raspberry Pi 3
more demo videos at
youtube.com/IdeinInc
Features

- Use of built-in VideoCore GPUs
  - no other accelerators and servers

- No model compression techniques (eg. quantization and pruning)
  - no accuracy drop
  - portable between devices
Benchmarks (Comparison with TensorFlow)

Inference speed with MobileNet v1 1.0 224 32-bit floating-point precision

- Both runs same model with same accuracy
- No extra hardware

<table>
<thead>
<tr>
<th>Device</th>
<th>Speed (images/sec)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>On Raspberry Pi Zero</td>
<td>8.33</td>
<td>49.0x</td>
</tr>
<tr>
<td></td>
<td>0.17</td>
<td></td>
</tr>
<tr>
<td>On Raspberry Pi 3</td>
<td>8.70</td>
<td>6.3x</td>
</tr>
<tr>
<td></td>
<td>1.39</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Resolution</td>
<td>Model</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>1000-class Image Classification</td>
<td>224x224</td>
<td>MobileNet V2 $\alpha=1.0$ trained on ImageNet dataset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Top-1: 71.8%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Top-5: 91.0%</td>
</tr>
<tr>
<td>Face Detection</td>
<td>192x192</td>
<td>Our custom model based on MobileNet V2 SSD trained on Open Images V4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mAP: 66.3%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi Person Pose Estimation</td>
<td>224x224</td>
<td>Our custom model based on Pose Proposal Network trained on MPII</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dataset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Style Transfer</td>
<td>256x256</td>
<td>Our custom model based on Fast Neural Style</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The speed of face detection and pose estimation depends on the number of objects in a frame. This benchmark is taken with one object.
# VideoCore GPU IV and VI

<table>
<thead>
<tr>
<th></th>
<th>BCM2835</th>
<th>BCM2837</th>
<th>BCM2711</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Device</strong></td>
<td>Pi Zero and so on</td>
<td>Pi 3 and so on</td>
<td>Pi 4</td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td>1GHz single-core ARM1176JZF-S</td>
<td>1.2GHz 64-bit quad-core ARM Cortex-A53</td>
<td>1.5GHz 64-bit quad-core ARM Cortex-A72</td>
</tr>
<tr>
<td><strong>GPU</strong></td>
<td>VideoCore IV</td>
<td>VideoCore VI</td>
<td></td>
</tr>
<tr>
<td><strong>CPU peak performance</strong></td>
<td>2Gflops</td>
<td>38.4Gflops</td>
<td>48Gflops</td>
</tr>
<tr>
<td><strong>GPU peak performance</strong></td>
<td>28.8Gflops</td>
<td>28.8Gflops</td>
<td>32Gflops</td>
</tr>
</tbody>
</table>
How to use VideoCore?

- We have developed software tools from scratch such as
  - Assembler for VC4
    - [github.com/nineties/py-videocore](https://github.com/nineties/py-videocore)
  - Assembler for VC6
    - [github.com/Idein/py-videocore6](https://github.com/Idein/py-videocore6)
- SW for DL is not fully open-sourced yet but math kernel libraries are open
  - [github.com/Idein/qmkl](https://github.com/Idein/qmkl)
  - [github.com/Idein/qmkl6](https://github.com/Idein/qmkl6)
nop(sig = ldtmu(MAP_4x18_REGS[1]))
mov(tmuA, B_CUR_REG),add(B_CUR_REG, B_CUR_REG, B_STRIDE_REGS[2])
nop(sig = ldtmu(MAP_4x18_REGS[2]))
mov(tmuA, B_CUR_REG),add(B_CUR_REG, B_CUR_REG, B_STRIDE_REGS[2])
mov(r3, 0, sig = ldtmu(MAP_4x18_REGS[3]))
mov(tmuA, B_CUR_REG),add(B_CUR_REG, B_CUR_REG, B_STRIDE_REGS[2])
eidx(r1, sig = ldtmu(MAP_4x18_REGS[0]))
mov(tmuA, B_CUR_REG),add(B_CUR_REG, B_CUR_REG, B_STRIDE_REGS[2])
mov(r0, MAP_4x18_REGS[0], sig = ldtmu(MAP_4x18_REGS[4]))
mov(tmuA, B_CUR_REG),add(B_CUR_REG, B_CUR_REG, B_STRIDE_REGS[2])

with loop as bi:

# mov(tmuA, B_CUR_REG) .mov(broadcast, r0, sig = ldtmu(K_REG[0][0]))
fadd(rs[1][15], rs[1][15], r3).fmlmul(r3, K_REG[0][0], r5, sig = ldtmu(K_REG[0][1]))
fadd(rs[0][0], rs[0][0], r3).rotate(broadcast, r0, -1)
mov(r0, MAP_4x18_REGS[1]) fmlmul(r3, K_REG[0][1], r5)
fadd(rs[0][0], rs[0][0], r3).fmlmul(r3, K_REG[0][0], r5)
shl(r1, r1, 2).mov(broadcast, A_CUR_REG)
add(r1, r1, r5) .mov(broadcast, r0, sig = ldtmu(K_REG[0][2]))
fadd(rs[0][1], rs[0][1], r3).fmlmul(r3, K_REG[0][2], r5)
fadd(rs[0][0], rs[0][0], r3).fmlmul(r3, K_REG[0][1], r5)
fadd(rs[0][1], rs[0][1], r3).fmlmul(r3, K_REG[0][0], r5)
fadd(rs[0][2], rs[0][2], r3).rotate(broadcast, r0, -1)
mov(tmuA, r1) .fmlmul(r3, K_REG[0][2], r5)
fadd(rs[0][1], rs[0][1], r3).fmlmul(r3, K_REG[0][1], r5)
fadd(rs[0][2], rs[0][2], r3).fmlmul(r3, K_REG[0][0], r5)
fadd(rs[0][3], rs[0][3], r3).rotate(broadcast, r0, -2)
add(r1, r1, A_STRIDE_REGS[2]) .fmlmul(r3, K_REG[0][2], r5)
fadd(rs[0][2], rs[0][2], r3).fmlmul(r3, K_REG[0][1], r5)
fadd(rs[0][3], rs[0][3], r3).fmlmul(r3, K_REG[0][0], r5)
fadd(rs[0][4], rs[0][4], r3).rotate(broadcast, r0, -3)
mov(tmuA, r1) .fmlmul(r3, K_REG[0][2], r5)
fadd(rs[0][3], rs[0][3], r3).fmlmul(r3, K_REG[0][1], r5)
Deep Learning Optimizing Compiler

- Optimizing Compiler dedicated for Machine Learning Models
  - Convert ONNX (or our format) to C
  - Can be used for any device in which C can be used
    - generated codes do not depend on any runtime library for specific environment
  - Adopted for in-vehicle SW (Misra-C output is supported)
- Use of hardware accelerators
  - VideoCore IV/VI
  - Intel AVX/SSE
  - ARM NEON
  - ARM Mali GPU
- Computational graph level optimization, automatic selection of convolutional algorithms, memory allocation and scheduling optimization, etc.
- Adopted an approach to speed up the process by using a highly efficient processor without using model compression techniques such as quantization and distillation that compromise accuracy.
  - Achieved higher performance than existing software on many processors
The SW pipeline

PyTorch → ONNX → ActDK

- Python Code + Schema
- CLI, Optimizing Compiler, AppFramework

Same model and code for edge devices that can run efficiently on a variety of hardware
Generate the app
Learn Once, Run Anywhere

- High-Spec device
- Low-Spec device
- High-Tolerance device

Idein Inc.
ONNX: Open Neural Network Exchange format

Major DL frameworks are supported. ActDK can be used with various DL frameworks.

Supported tools (https://onnx.ai)

ONNX partners (https://onnx.ai)
Architecture of the compiler

Deep learning frameworks

- TensorFlow
- Chainer

Model Converters
- ONNX
- nnoir

Computation Graph Representation

Graph Compiler
- Pure-GPU Code Generation
- Neural Network API
- libnn
- qdnn

runtime library

Writing CNN kernels
- Doing tests and optimization

py-video

Designed and Developed by Idein Inc.

VC4 | AVX/SSE | NEON | CPU | Mobile GPU

Idein Inc.
Architecture of VideoCore IV

- simple SIMD architecture
- 4x3 16way SIMD core
  - fp32 accumulation and multiplication
dual issue
  - Special function unit
- No dynamic register allocation and scheduling
- Low memory bandwidth

Using Raspberry Pi GPU for Deep Neural Networkより
QPU / Quad Processing Unit

Register

| f32 |

ALU Instruction


TMU / Texture and Memory Lookup Unit

GPU memory

L2 Cache

TMU0  TMU1

Store r4 Register

Load Signal

Idein Inc.
Inner/Direct Product

Direct Product Base

TMU0

TMU1

Inner Product Base

×

→

×

→

×

→

×

→
Better bandwidth utilization
Optimization of DL for low-end mobile GPU

- Algorithm Selection
- Tensor Layout Selection
- Classic Loop Optimizations
- Layer Merging
- Layer Level Scheduling
- and others
Algorithm Selection

Example

- algorithms for 2D Convolution
- input size: HxW, file size=3x3

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Number of operations</th>
<th>Memory Utilization</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>x1</td>
<td>input: x1, weight: x1</td>
<td>Rarely used but we mostly use this for VC4/6.</td>
</tr>
<tr>
<td>im2col</td>
<td>x1</td>
<td>input: x9, weight: x1</td>
<td>Most popular</td>
</tr>
<tr>
<td>winograd</td>
<td>x4/9</td>
<td>input: x1, weight: x16/9</td>
<td>Popular. Not good fit for Separable Convolution</td>
</tr>
<tr>
<td>FFT</td>
<td>x2/9 at most</td>
<td>input: x1, weight: xHW/9</td>
<td>Rarely used. Need complex arithmetic</td>
</tr>
</tbody>
</table>
Tensor Layout

- **CHW**
  - 1 2 3 4 5 6 7 8

- **HWC**
  - 1 2 3 4 5 6 7 8

- **HCW**
  - 1 2 3 4 5 6 7 8

---

- **Idein Inc.**

---

 japane symbols not translated
Classic Loop optimizations

- Loop Unrolling, Loop Interchange, Software Pipelining and so on
- Considering
  - order of memory access
  - alignment of the access
  - register utilization
  - pipeline stall
  - and so on
Layer Merging

```
Convolution
    └── Batch Normalization

Convolution
    └── ReLU
```

```
Convolution

Convolution + ReLU
```
Handy-tuned Specialized Convolution Kernels with these optimizations

- add
- average_pooling_2d
- batch_normalization
- conv2d
- deconv2d
- depthwise_conv2d
- div

- k1x1/sXxX/NCHW
- k1xX/sXxX/NCHW
- k3x3/s1x1/NCHW
- kXx1/sXxX/NCHW
- kXxX/sXxX/NCHW

- to_NCHW_H2xW10xC32.py
- to_NCHW_H2xW16xC16.py
- to_NCHW_H2xW16xC16_overlap_w.py
- to_NCHW_H4xW4xC32_Wfp16.py
- to_NCHW_H4xW8xC16.py
- to_NHWC_H2xW10xC32.py
- to_NHWC_H2xW16xC16.py
- to_NHWC_H2xW16xC16_overlap_w.py
- to_NHWC_H4xW4xC32_Wfp16.py
- to_NHWC_H4xW8xC16.py
- to_NHWC_H2xW10xC32.py
- to_NHWC_H2xW16xC16.py
- to_NHWC_H2xW16xC16_overlap_w.py
- to_NHWC_H4xW8xC16.py
Layer Level Scheduling

Determine the order of execution and data allocation to minimize memory consumption by solving combinatorial optimization problems.

<table>
<thead>
<tr>
<th></th>
<th>Before</th>
<th>After</th>
<th>% Decrease</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNet V2</td>
<td>13.3MB</td>
<td>9.3MB</td>
<td>30%↓</td>
</tr>
<tr>
<td>ShuffleNet</td>
<td>5.0MB</td>
<td>3.0MB</td>
<td>40%↓</td>
</tr>
</tbody>
</table>

Visualization of Memory Usage Scheduling
Others: Removing ioctl overhead

- Overhead of ioctl() call is too big in case of Pi Zero
  - ioctl() is called for each layer
- Moved control-flow management from CPU to GPU for removing this overhead.
  - VC4/6 has not hardware mechanisms for that such as stack memory, link register and so on.
  - Our compiler simulates and resolve all control-flow and memory allocation ahead of time
Summary

- Raspberry Pi is popular, low-price, versatile and has a big ecosystem in terms of users, companies, software and hardware.
- We showed that Raspberry Pi has enough computing power to run various deep learning inference at a practical speed and accuracy.
- Maximizing Memory Utilization is the most important for DL on low-end mobile GPUs.
Arm: The Software and Hardware Foundation for tinyML

1. Connect to high-level frameworks
2. Supported by end-to-end tooling
3. Connect to Runtime

Profiling and debugging tooling such as Arm Keil MDK

Optimized models for embedded

Runtime (e.g. TensorFlow Lite Micro)

Optimized low-level NN libraries (i.e. CMSIS-NN)

RTOS such as Mbed OS

Arm Cortex-M CPUs and microNPUs

Stay Connected

@ArmSoftwareDevelopers

@ArmSoftwareDev

Resources: developer.arm.com/solutions/machine-learning-on-arm
Dynamic Neural Accelerator™

- 10x more compute with single DNA engine
- More than 20x better energy-efficiency
- Ultra-low latency
- Fully-programmable with INT 8bit support

TARGET MARKETS
- Automotive
- Robotics
- Drones
- Smart Cities
- Industry 4.0

Tight coupling between software & hardware with automated co-design

www.edgecortix.com

© 2020 EDGECORTIX. ALL RIGHTS RESERVED
SynSense builds ultra-low-power (sub-mW) sensing and inference hardware for embedded, mobile and edge devices. We design systems for real-time always-on smart sensing, for audio, vision, IMUs, bio-signals and more.

https://SynSense.ai
Questions?
Or to join tinyML WeChat Group

Please add staff to join our official tinyML WeChat Group
Copyright Notice

This presentation in this publication was presented at tinyML® Asia 2020. The content reflects the opinion of the author(s) and their respective companies. The inclusion of presentations in this publication does not constitute an endorsement by tinyML Foundation or the sponsors.

There is no copyright protection claimed by this publication. However, each presentation is the work of the authors and their respective companies and may contain copyrighted material. As such, it is strongly encouraged that any use reflect proper acknowledgement to the appropriate source. Any questions regarding the use of any materials presented should be directed to the author(s) or their companies.

tinyML is a registered trademark of the tinyML Foundation.

www.tinyML.org
User-friendly and low-price PaaS (Platform as a Service) for Solution vendors/End users to create own IoT system with advanced AI analysis

Edge Computing Architecture enables significant cost reduction and privacy protection

https://actcast.io  SDK is available for Free. Join our partner program!