How to Build Optimized ML Applications with Arm Software

Arm Technical Symposia 2018

Arm K.K. Senior FAE
Ryuji Tanaka

© 2018 Arm Limited
Overview

Today we will talk about applied machine learning (ML) on Arm. My aim for today is to show you just how much can be achieved on mobile and edge devices.

We will cover:

• An overview of the ML platform
• A look at some use cases
• The software and tools we use
• Our approach to development and deployment
Solving Problems
Use Case Fragments

Combining these pieces helps to solve larger problems

Image Processing
- Noise reduction and super resolution for better video quality
- Foreground/background segmentation for live video and camera stills

Speech and Audio
- Keyword and noise spotting (KWS) in low power for system wakeup
- Speech recognition, translation, text to speech (TTS)

Vision
- Object detection for autonomous driving, mobile or security
- Identifying points of interest for augmented reality mapping

User Interface
- Natural language processing (NLP) to mine text and improve UIs
- Improving OS power management and scheduling
Face ID and Unlock

- Arm Cortex-M7 processes input from a low-resolution sensor using an NN in an always-on power budget.
- This object detection gates the processing of RGBD camera data that runs on the Cortex-A73.
- Higher quality *face detection* removes false positives, and *spoof detection* uses depth information to make sure the face isn’t just a photograph.
- The identity verification stage is a neural network designed to highlight differentiable features in faces.
- Moving stages to the right IP is natural when using Arm NN and Compute Library.
Face ID

- Face ID is a key technology being deployed in phones, homes and cities
- It enables unlocking of phones, making payments, logging in and opening doors

This use case is tuned for CPU, GPU and NPU

*8-core Mali-G71 silicon @850MHz and Arm NPU @1GHz
Babelfish
Automatic Speech Recognition
Going Fishing

Why look at the Babelfish problem?

- NN ASR for edge is moving out of the research space
  - Now more accurate than humans on word-error rates
- ASR, translation and generation are very complex
  - Speech generation is more complex than vision in some cases
- This needs more than just CPU and GPU
  - Complex workloads start to need Terra-OP class hardware
- ASR on the edge is good for users
  - Latency is better, privacy is maintained for sensitive conversations
  - Combining ASR and vision is compelling
Real-Time Subtitles (Augmented Reality)

Assist deaf people by adding a virtual layer of real-time subtitles to their view

**Who**
Identify the speaker’s ID using a speaker-recognition system

**Where**
Using visual face-identification system we find the bounding box of that speaker

**What**
Automatic speech recognition produces a transcript from the audio feed

---

Speaker Identification

Face Identification

Voice and Image Associated

Speech Recognition

“Let’s go”
No! So what did you do?
No! So what did you do?

I just kept going and hoped for the best!
No! So what did you do?

Brilliant! A confident presenter makes all the -

I just kept going and hoped for the best!

You mean they didn’t notice it was -
No! So what did you do?

Brilliant! A confident presenter makes all the -

Real-time, on-device speech recognition

60 FPS face detection and audio-visual speaker recognition
How Did We Deploy This Use Case?
# Project Trillium: Arm’s ML Computing Platform

## Ecosystem
- TensorFlow
- Caffe
- Caffe2
- mxnet
- Android NNAPI

## Software Libraries Optimized for Arm Hardware
- CMSIS-NN
- armNN
- arm COMPUTE LIBRARY
- Object Detection Libraries

## Hardware Products

### CPU
- arm CORTEX-A
- arm NEON
- arm CORTEX-M
- armv8 SVE
- arm DynamIQ

### GPU
- arm MALI

### NPU and ODP
- Machine Learning (ML) Object Detection (OD) processors

### Partner IP
- DSPs, FPGAs, Accelerators
What is Arm NN?

An inference engine for edge machine learning

1. Program through standard ML Frameworks
2. Use accelerated support on Android
3. Designed to support Arm ML Processor and third-party IP

Key Arm NN aims

- Well optimized for Arm CPUs, GPUs and NPUs
- Interoperation with other inference engines
- Low overhead for embedded systems
Collaborate to Improve Standard ML Software Interfaces

• Arm has donated Arm NN to Linaro to improve common software interface for ML
• Starting contributing code directly and help us add exciting new features
• Join Linaro Machine Learning Initiative

Domain experts | Machine learning experts | Platform developers
Tools for Optimization and Conditioning

Conditioning

• Quantizing and retraining networks from FP32 to INT8 without accuracy loss

• Compression of weights and reduction of operations by pruning and clustering weights

Optimizing

• Streamline adds support for Arm NN in addition to the GPU driver to show end-to-end timing

• System reports provide feedback on what should be optimized
Quantization to 8-bit: It Works
Theory for CNN and practical deployment possible today

<table>
<thead>
<tr>
<th>Network (Recognition)</th>
<th>FP32 Top-1 Accuracy</th>
<th>FP32 Top-5 Accuracy</th>
<th>UINT8 Top-1 Accuracy</th>
<th>UINT8 Top-5 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG 16 (ImageNet)</td>
<td>69.88%</td>
<td>89.16%</td>
<td>68.39%</td>
<td>88.26%</td>
</tr>
<tr>
<td>VGG 19 (ImageNet)</td>
<td>70.05%</td>
<td>89.17%</td>
<td>68.65%</td>
<td>88.31%</td>
</tr>
<tr>
<td>ResNet V2 101 (ImageNet)</td>
<td>70.50%</td>
<td>89.75%</td>
<td>66.56%</td>
<td>86.01%</td>
</tr>
<tr>
<td>ResNet V2 152 (ImageNet)</td>
<td>71.04%</td>
<td>90.04%</td>
<td>68.93%</td>
<td>86.70%</td>
</tr>
<tr>
<td>Inception V3 (ImageNet)</td>
<td>76.07%</td>
<td>92.59%</td>
<td>74.60%</td>
<td>91.78%</td>
</tr>
<tr>
<td>Inception V4 (ImageNet)</td>
<td>78.33%</td>
<td>93.93%</td>
<td>77.07%</td>
<td>93.29%</td>
</tr>
<tr>
<td>MobileNet 1.0 224</td>
<td>69.60%</td>
<td>89.12%</td>
<td>62.75%</td>
<td>84.25%</td>
</tr>
<tr>
<td>MobileNet 1.0 224 - retrained</td>
<td>70.90%</td>
<td>89.90%</td>
<td>69.70%</td>
<td>89.50%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network (Segmentation)</th>
<th>FP32 – Pixel Accuracy</th>
<th>FP32 – F1-Score</th>
<th>UINT8 – Pixel Accuracy</th>
<th>UINT8 – F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully-connected Networks (KITTI)</td>
<td>94.15%</td>
<td>72.30%</td>
<td>95.61%</td>
<td>84.79%</td>
</tr>
<tr>
<td>Fully-connected Networks (VOC 2011)</td>
<td>85.96%</td>
<td>50.86%</td>
<td>85.92%</td>
<td>50.48%</td>
</tr>
</tbody>
</table>

Source: Arm ML Technology group research using custom techniques – [updated MobileNet retraining from Google](https://www.arm.com)
Distillation of Speech Networks
Teaching a small convolutional network from a large recurrent one

What is distillation?

Selected student and teacher networks

- Teacher: Large RNN
- Student: Small CNN
- Soft Logits
- Optimize Joint Loss

- wav2letter
- DeepSpeech
- Baidu Pre-Trained Model
- 960 hrs data
- 8528 hrs data
- CNN ~ 20M parameter
- RNN ~ 120M parameters

Training Data (Libri-Speech)
Distillation of Speech Networks
Teaching a small convolutional network from a large recurrent one

What is distillation?

Result: better accuracy and generalization with headroom for further improvement

Letter Error Rates (No Language Model)

- Original Paper Wav2Letter
- Our Retrained Wav2Letter
- Baidu's DeepSpeech 2

Standard wav2letter
Distilled wav2letter
# Performance Results

<table>
<thead>
<tr>
<th>Pipeline Stage</th>
<th>Operations Per Inference on 1 Second/1 Frame/Letter</th>
<th>Inferences per Second of Realtime Input</th>
<th>Operations per Second Required</th>
<th>Mid-end 2017 Mobile (ms per second of input)*</th>
<th>Complex Version Processing Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Extraction</td>
<td>0.0032 GOP/s</td>
<td>138 samples per second</td>
<td>0.44 GOP/s</td>
<td>24 ms</td>
<td>24 ms</td>
</tr>
<tr>
<td>Acoustic Model</td>
<td>7.3 GOP/ss Wav2Letter</td>
<td>4 due to overlapping 1 second window with 0.25 second stride</td>
<td>29.2 GOP/s</td>
<td>200 ms</td>
<td>760 ms (small accuracy improvements)</td>
</tr>
<tr>
<td>Language Model</td>
<td>Ken LM (Mozilla/Baidu) 10 – 800 GOP/s</td>
<td>0.33 as the language model works on a 3 second window</td>
<td>3.3 – 264 GOP/s</td>
<td>91 ms</td>
<td>2380 ms (more languages)</td>
</tr>
<tr>
<td>Translation / NLP</td>
<td>124 – 562 MOP/s</td>
<td>15</td>
<td>1.9 – 8.4 GOP/s</td>
<td>20 ms</td>
<td>88 ms (accuracy improvements)</td>
</tr>
<tr>
<td>Face Detection</td>
<td>5.4 – 65 GOP/s</td>
<td>30 – 60 FPS</td>
<td>162 GOPs – 3.9 TOP/s</td>
<td>30 ms</td>
<td>200 ms (accuracy improvements)</td>
</tr>
<tr>
<td>Face Verification</td>
<td>8.46 GOP/s</td>
<td>3 FPS</td>
<td>12.3 GOP/s</td>
<td>442 ms</td>
<td>442 ms</td>
</tr>
<tr>
<td>Speaker ID</td>
<td>20.28 MOP/s</td>
<td>1 one sample a second</td>
<td>20.28 MOP/s</td>
<td>10 ms</td>
<td>10 ms</td>
</tr>
<tr>
<td>Speech Generation</td>
<td>0.1 – 1.3 TOP/s</td>
<td>22 Khz</td>
<td>0.1 – 1.3 TOP/s</td>
<td>280 ms</td>
<td>3250 ms (more realistic speech)</td>
</tr>
<tr>
<td>Combined</td>
<td></td>
<td></td>
<td>0.4 – 5.9 TOP/s</td>
<td>809 ms</td>
<td>6146 ms</td>
</tr>
</tbody>
</table>

* Measured on the Arm NN 18.05 release – notable optimizations have been introduced since
Conclusion

- This is possible today on production devices
  - On today’s platforms using CPUs and GPUs
  - And Arm NPU platforms appearing over next year
- A lot can be achieved on the shoulders of giants
  - The ML research community is not just for huge companies
  - Retraining with distillation is a key tool
  - We provide tool flows to extract sizable performance gains
- We’re continuing to work to make Babelfish a reality
  - We’re not far from this aim, the next year or two will be exciting
Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!
감사합니다
धन्यवाद