Machine Learning on Ultra-Constrained Devices

Mark Woods
Solutions Architect
Flexible, Scalable ML Solutions

Only Arm can enable ML everywhere

Deliver use cases with multiple hardware solutions
Choose best balance of ML performance versus capabilities per use case
Why ML is Moving to the Edge

- Bandwidth
- Power
- Cost
- Latency
- Reliability
- Security
Machine Learning and AI Basics

**Artificial Intelligence:** Umbrella term for machines acting like they are ‘thinking’

**Machine Learning:** machines adapting algorithms based on experience (e.g. labelled pictures)

**Deep Learning:** machine learning using Deep Neural Network approaches

**Algorithms:** CNNs, RNNs, etc.

---

Training data

Neural network

Model

Inference

New input

Neural network w/model

97.4% confidence
Project Trillium: Arm’s ML Computing Platform

AI/ML Applications, Algorithms and Frameworks

- TensorFlow
- TensorFlow Lite
- PyTorch
- ONNX
- Caffe
- Caffe2
- Android NNAPI

Software Libraries Optimized for Arm Hardware

- armNN
- arm Compute Library
- CMSIS-NN

Arm Hardware IP for AI/ML

- CPU: arm CORTEX, arm DynamIQ, arm NEOVERSE
- GPU: arm MALI
- NPU: Machine Learning Processor (ML)
- Partner IP: DSPs, FPGAs, Accelerators
Arm’s Heterogeneous Machine Learning Platform

AI applications
Incorporating ML, CV, speech recognition etc.

Higher-level libraries

Neural network frameworks
(e.g. TensorFlow, Caffe, Android NNAPI) & third-party tools

armNN

Compilers/Drivers

Hardware

CPU
- CORTEX-M
- Arm Helium

CPU
- CORTEX-A
- NEON
- DynamIQ

GPU
- arm Mali

NPU
- Neural Processing Unit
- Machine Learning processor

Partner IP
- DSPs, FPGAs, accelerators

Provided by third party

Provided by Arm

Stable SW interfaces
Collaborate to Improve Standard ML Software Interface

• Arm donated Arm NN to Linaro to accelerate development of a common software interface for ML
• Ways to help:
  • Start contributing code
  • Help us add exciting new features
  • Join Linaro Machine Learning Initiative

• developer.arm.com/arm-nn
Cortex Microcontroller Software Interface Standard (CMSIS)

- CMSIS: vendor-independent hardware abstraction layer for Cortex-M CPUs
- Enables consistent device support, reduce learning curve and time-to-market
- Open-source: [https://github.com/ARM-software/CMSIS_5/](https://github.com/ARM-software/CMSIS_5/)
CMSIS-NN

- Optimized low-level NN functions for Cortex-M CPUs
- A collection of efficient neural network kernels developed to maximize the performance and minimize the memory footprint of neural networks on Cortex-M processor cores
- Publicly available now (no fee, Apache 2.0 license)

https://developer.arm.com/embedded/cmsis

NN Runtime improvement
4.6x higher perf.

Energy efficiency improvement
4.9x higher eff.

Application
TensorFlow / Caffe etc.
Arm NN in development

CMSIS-NN
Cortex-M
Arm Cortex-M processor portfolio

Cortex-M7
- Maximum performance, control and DSP

Cortex-M0
- Lowest cost, low power

Cortex-M0+
- Highest energy efficiency

Cortex-M3
- Performance efficiency

Cortex-M33
- Flexibility, control and DSP

Cortex-M35P
- Tamper resistance, flexibility, control and DSP

Cortex-M4
- Mainstream control and DSP

Cortex-M23
- Smallest area, lowest power

Armv6-M

Armv7-M

Armv8-M

TrustZone

High performance

Performance efficiency

Lowest power & area
Developing NN Solutions on Cortex-M

Problem

Hardware-constrained NN model search

Trained NN model

NN model translation

Deployable solution

Optimized NN functions: CMSIS-NN

Optimized code on h/w
Developing NN Solutions on Cortex-M

- Problem
- Trained NN model

Hardware-constrained NN model search

- NN model translation
- Deployable solution
- Optimized NN functions: CMSIS-NN
- Optimized code on h/w
Use Case – Keyword Spotting

• Listen for certain words / phrases
  • Voice activation: “Ok Google”, “Hey Siri”, “Alexa”
  • Simple commands: “Play music”, “Pause”, “Set Alarm for 10 am”

Feature extraction

- FFT-based mel frequency cepstral coefficients (MFCC) or log filter bank energies (LFBE)

Classification

- Neural network based – DNN, CNN, RNN (LSTM/GRU) or a mix of them
Arm Cortex-M based MCU Platforms

Small
- **ST Nucleo-F103RB**
  - Cortex-M3, 72 MHz, 20KB RAM, 128KB Flash
- **Nordic nRF52-DK**
  - Cortex-M4, 64 MHz, 64KB RAM, 512KB Flash

Medium
- **ST Nucleo-F411RE**
  - Cortex-M4, 100 MHz, 128KB RAM, 512KB Flash
- **ST Nucleo-F746ZG**
  - Cortex-M7, 216 MHz, 320KB RAM, 1MB Flash

Large
- **NXP i.MX RT 1050**
  - Cortex-M7, 600 MHz, 512KB RAM

More boards at: [https://os.mbed.com/platforms](https://os.mbed.com/platforms)
Keyword Spotting on NN Models: Memory vs. Ops

- Need **compact models**: that fit within the Cortex-M system memory
- Need models with **less operations**: to achieve real time performance
- Neural network model search parameters
  - NN architecture
  - Number of input features
  - Number of layers (3-layers, 4-layers, etc.)
  - Types of layers (conv, ds-conv, fc, pool, etc.)
  - Number of features per layer

NN Models from literature trained on Google speech commands dataset
Depthwise Separable CNN (DS-CNN) achieves highest accuracy

Accuracy asymptotically reaches to 95%.

Training scripts and models are available on github:
https://github.com/ARM-software/ML-KWS-for-MCU
Developing NN Solutions on Cortex-M

Problem

Hardware-constrained NN model search

Trained NN model

NN model translation

Deployable solution

Optimized NN functions: CMSIS-NN

Optimized code on h/w
NN Model Quantization

• Quantization type impacts model size and performance
  • Bit-width: 8-bit or 16-bit
  • Symmetric around zero or not
  • Quantization range as \([-2^n, 2^n)\) a.k.a. fixed-point quantization

• Steps in model quantization
  • Run quantization sweeps to identify optimal quantization strategy
  • Quantize weights: does not need a dataset
  • Quantize activations: run the model with dataset to extract ranges

Neural networks can tolerate quantization

<table>
<thead>
<tr>
<th>NN model</th>
<th>32-bit floating point model accuracy</th>
<th>8-bit quantized model accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Val.</td>
</tr>
<tr>
<td>DNN</td>
<td>97.77%</td>
<td>88.04%</td>
</tr>
<tr>
<td>Basic LSTM</td>
<td>98.38%</td>
<td>92.69%</td>
</tr>
<tr>
<td>GRU</td>
<td>99.23%</td>
<td>93.92%</td>
</tr>
<tr>
<td>CRNN</td>
<td>98.34%</td>
<td>93.99%</td>
</tr>
</tbody>
</table>

• Minimal loss in accuracy (~0.1%).
• May increase accuracy in some cases
  • Regularization (or reduced over-fitting)
Developing NN Solutions on Cortex-M

Problem

Hardware-constrained NN model search

Trained NN model

NN model translation

Deployable solution

Optimized NN functions: CMSIS-NN

Optimized code on h/w

Developing NN Solutions on Cortex-M

© 2019 Arm Limited
Demo - Multiple Neural Networks on Cortex-M7

Both image classification and keyword spotting are running at the same time

Voice command controls the start/stop of the image classification

CNN: 87 kB weights + 40 kB activations + 10 kB buffers
DNN: 98 kB weights + 1 kB activations + 2 kB buffers
A Convolutional Neural Network Example

- CIFAR-10 classification – classify images into 10 different object classes
  - 50k training images and 10k test images
- CNN Example from Caffe
  - 3 convolution layer, 3 pooling layer and 1 fully-connected layer
  - Trained in Caffe with ~80% fp32 accuracy

CIFAR-10 dataset: [https://www.cs.toronto.edu/~kriz/cifar.html](https://www.cs.toronto.edu/~kriz/cifar.html)
Convolutional Neural Network (CNN) on Cortex-M7

- CNN with 8-bit weights and 8-bit activations
- Total memory footprint: 87 kB weights + 40 kB activations + 10 kB buffers (I/O etc.)
- Example code available in CMSIS-NN github

<table>
<thead>
<tr>
<th>Layer</th>
<th>Network Parameter</th>
<th>Output activation</th>
<th>Operation count</th>
<th>Runtime on M7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv1</td>
<td>5x5x3x32 (2.3 KB)</td>
<td>32x32x32 (32 KB)</td>
<td>4.9 M</td>
<td>31.4 ms</td>
</tr>
<tr>
<td>Pool1</td>
<td>3x3, stride of 2</td>
<td>16x16x32 (8 KB)</td>
<td>73.7 K</td>
<td>1.6 ms</td>
</tr>
<tr>
<td>Conv2</td>
<td>5x5x32x32 (25 KB)</td>
<td>16x16x32 (8 KB)</td>
<td>13.1 M</td>
<td>42.8 ms</td>
</tr>
<tr>
<td>Pool2</td>
<td>3x3, stride of 2</td>
<td>8x8x32 (2 KB)</td>
<td>18.4 K</td>
<td>0.4 ms</td>
</tr>
<tr>
<td>Conv3</td>
<td>5x5x32x64 (50 KB)</td>
<td>8x8x64 (4 KB)</td>
<td>6.6 M</td>
<td>22.6 ms</td>
</tr>
<tr>
<td>Pool3</td>
<td>3x3, stride of 2</td>
<td>4x4x64 (1 KB)</td>
<td>9.2 K</td>
<td>0.2 ms</td>
</tr>
<tr>
<td>ip1</td>
<td>4x4x64x10 (10 KB)</td>
<td>10</td>
<td>20 K</td>
<td>0.1 ms</td>
</tr>
<tr>
<td>Total</td>
<td>87 KB weights</td>
<td>Total: 55 KB</td>
<td>24.7 M Ops</td>
<td>99.1 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Max. footprint: 40 KB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Convolutional Neural Network (CNN) on Cortex-M7

OpenMV.io Cam with a Cortex-M7

Video: https://www.youtube.com/watch?v=PdWi_fvY9Og
Keyword Spotting on Cortex-M7

- Speech Command dataset from Google
- Classifying the audio into: [“yes”, “no”, “up”, “down”, “left”, “right”, “on”, “off”, “stop”, “go”, silence, unknown]
- Network trained in Tensorflow: https://www.tensorflow.org/versions/master/tutorials/audio_recognition

Table: Best networks that can fit into Cortex-M4 and Cortex-M7

<table>
<thead>
<tr>
<th></th>
<th>Cortex-M4</th>
<th></th>
<th></th>
<th>Cortex-M7</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Memory</td>
<td>Ops</td>
<td>Accuracy</td>
<td>Memory</td>
<td>Ops</td>
</tr>
<tr>
<td>DNN</td>
<td>85.4 %</td>
<td>67.0K</td>
<td>133.0K</td>
<td>85.4 %</td>
<td>98.0K</td>
<td>194.4K</td>
</tr>
<tr>
<td>Basic LSTM</td>
<td>91.9 %</td>
<td>43.1K</td>
<td>4.0M</td>
<td>92.5 %</td>
<td>79.3K</td>
<td>7.4M</td>
</tr>
<tr>
<td>LSTM</td>
<td>91.7%</td>
<td>54.5K</td>
<td>5.1M</td>
<td>92.4%</td>
<td>96.2K</td>
<td>9.1M</td>
</tr>
<tr>
<td>GRU</td>
<td>92.4%</td>
<td>60.1K</td>
<td>5.6M</td>
<td>92.4%</td>
<td>60.1K</td>
<td>5.6M</td>
</tr>
<tr>
<td>CNN-GRU</td>
<td>92.2%</td>
<td>66.8K</td>
<td>2.5M</td>
<td>94.1%</td>
<td>196.3K</td>
<td>7.8M</td>
</tr>
<tr>
<td>Depthwise CNN</td>
<td>92.2%</td>
<td>55.1K</td>
<td>5.9M</td>
<td>93.8%</td>
<td>178.5K</td>
<td>16.4M</td>
</tr>
</tbody>
</table>
Machine Learning Model Deployment on Arm Cortex-M

Tensorflow/Caffe trained model → Dataset → Quantization → Code generation

NN-kernels optimized for Cortex-M based devices → weights.h, net.cpp → Compile (mbed-cli)

Cortex-M based platform (e.g. Nucleo-f411re) → Cortex-M class binary
Get started today with a wealth of resources from Arm

- CMSIS-NN Github link: [https://github.com/ARM-software/CMSIS_5/](https://github.com/ARM-software/CMSIS_5/)

- KWS blog: [https://community.arm.com/processors/b/blog/posts/high-accuracy-keyword-spotting-on-cortex-m-processors](https://community.arm.com/processors/b/blog/posts/high-accuracy-keyword-spotting-on-cortex-m-processors)
- KWS Github link: [https://github.com/ARM-software/ML-KWS-for-MCU/](https://github.com/ARM-software/ML-KWS-for-MCU/)

- ArmNN SDK blog: [https://community.arm.com/tools/b/blog/posts/arm-nn-sdk](https://community.arm.com/tools/b/blog/posts/arm-nn-sdk)
Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكر