Deep Learning on Arm Cortex-M Microcontrollers

arm

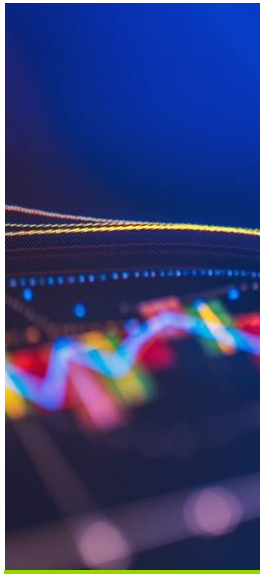
Vikas Chandra

May 23, 2018

Why is ML Moving to the Edge?

Bandwidth

Power



Cost

Latency

Reliability

Security

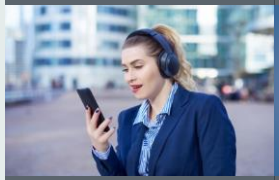
More Intelligence at the Edge

Arm Cortex-M

Expanding opportunity for the embedded intelligence market


Sensor fusion

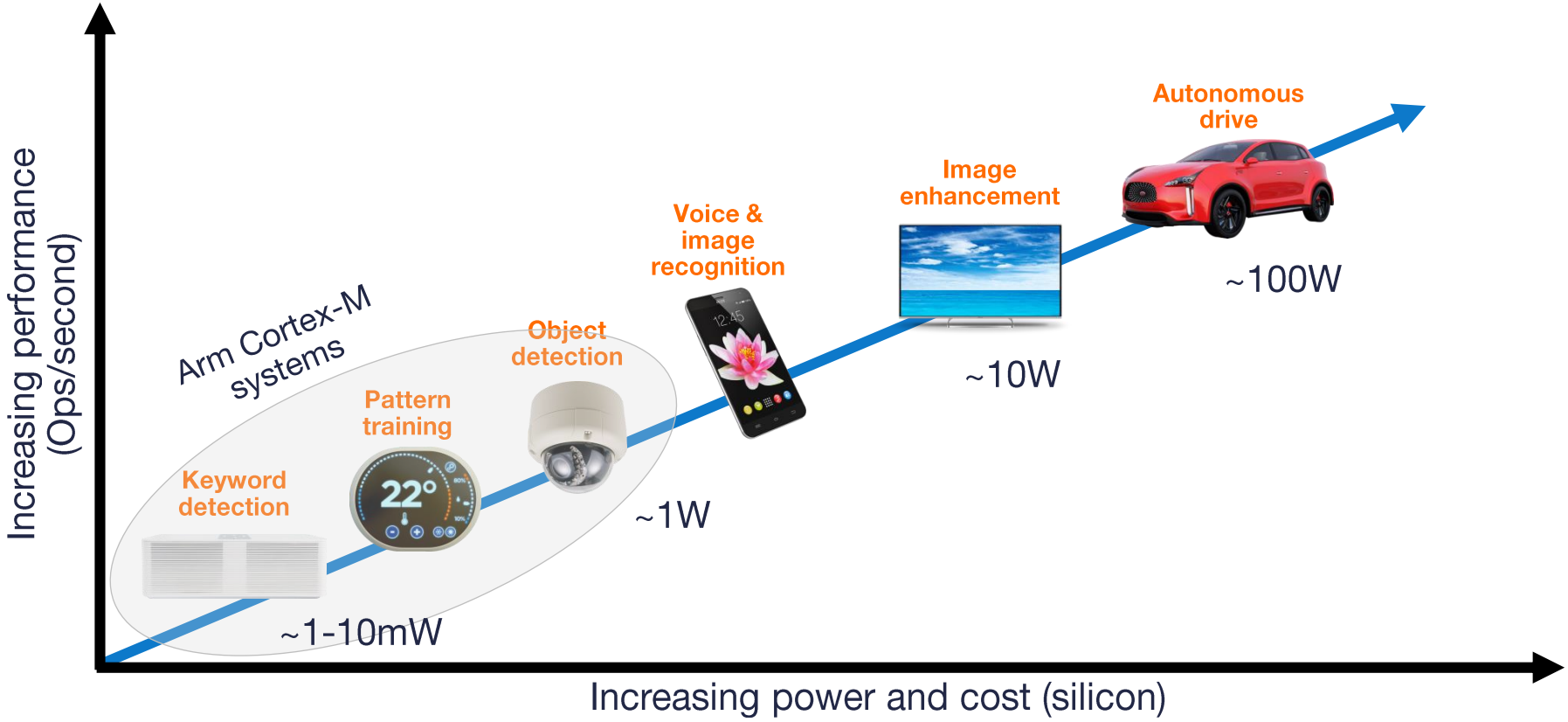

Contextual awareness


Voice activation

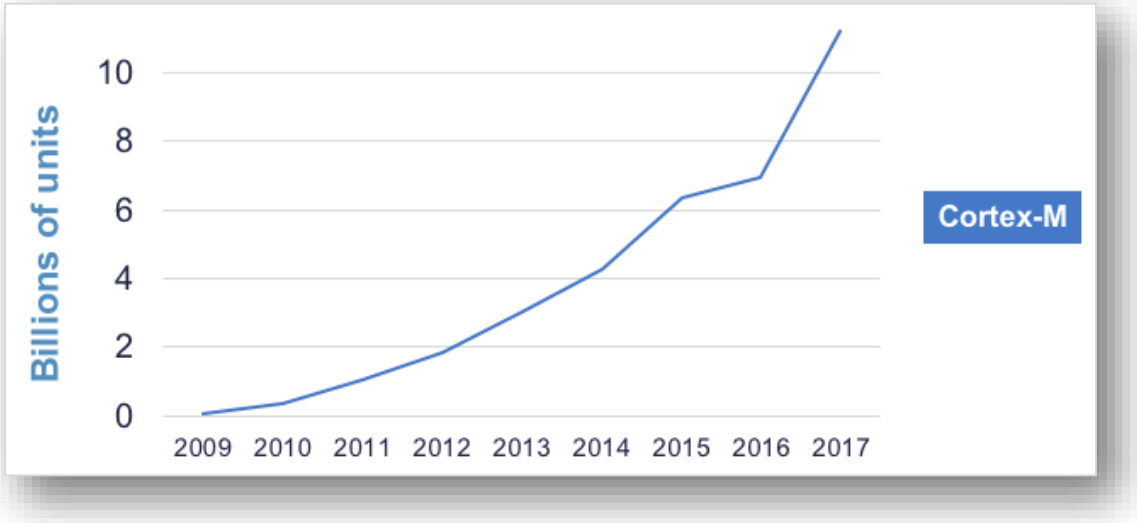

Biometrics

More data → More processing → More algorithms → More compute

Range of “Edge” Applications

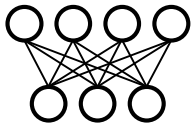


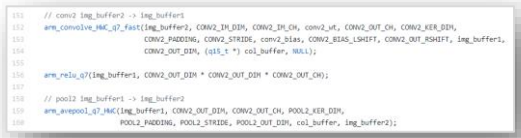
Size of Cortex-M Market



- **35 billion** Cortex-M based chips shipped to date!
- Enormous opportunity to enable deep learning on these devices

Developing NN Solutions on Cortex-M





Problem

Trained NN model

Deployable solution

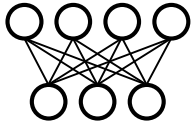
Optimized code on h/w

Hardware-constrained NN model search

NN model translation

Optimized NN functions: CMSIS-NN

Developing NN Solutions on Cortex-M



Problem

Trained NN model

Hardware-constrained NN model search

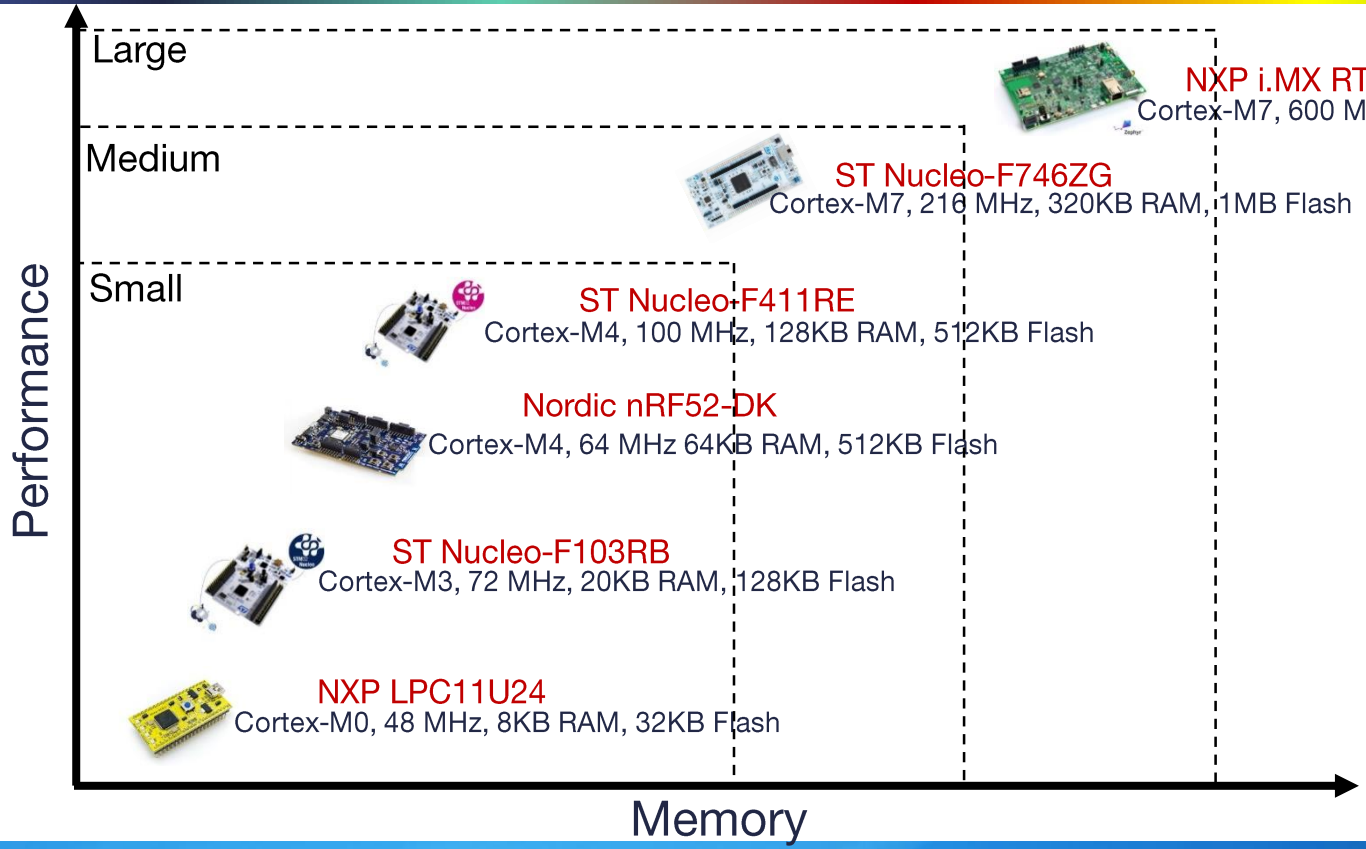
NN model translation

Deployable solution

Optimized NN functions: CMSIS-NN

Optimized code on h/w

Arm Cortex-M based MCU Platforms



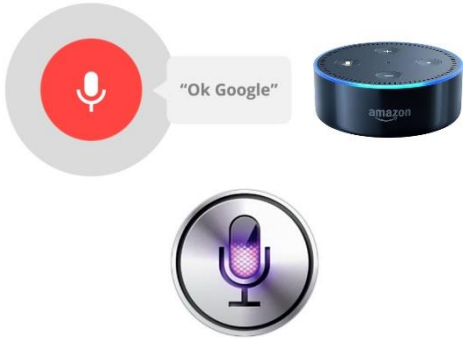
More boards at: https://os.mbed.com/platforms/

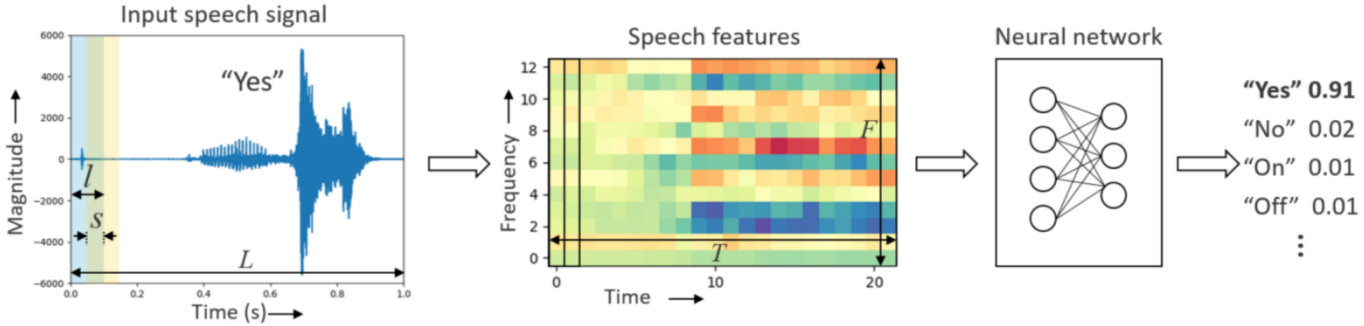
NN Models: Memory vs. Ops

- 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

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”





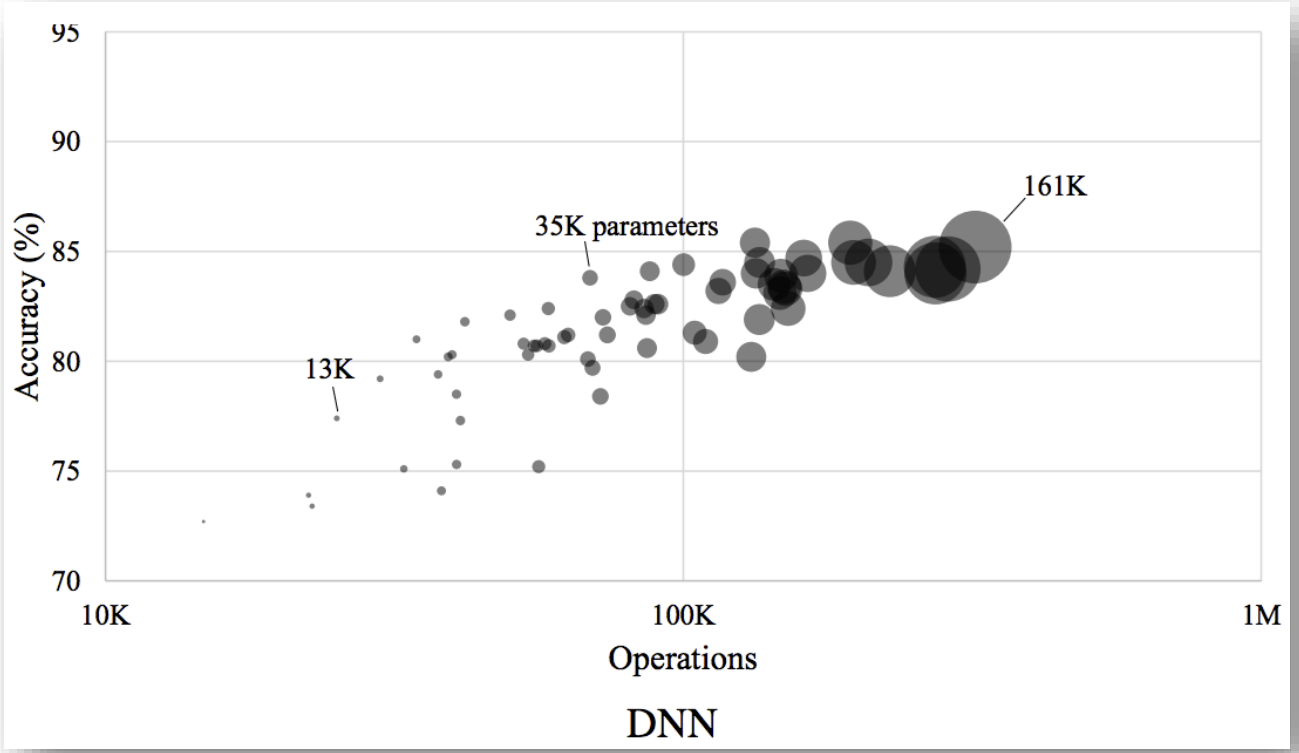
Feature extraction

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

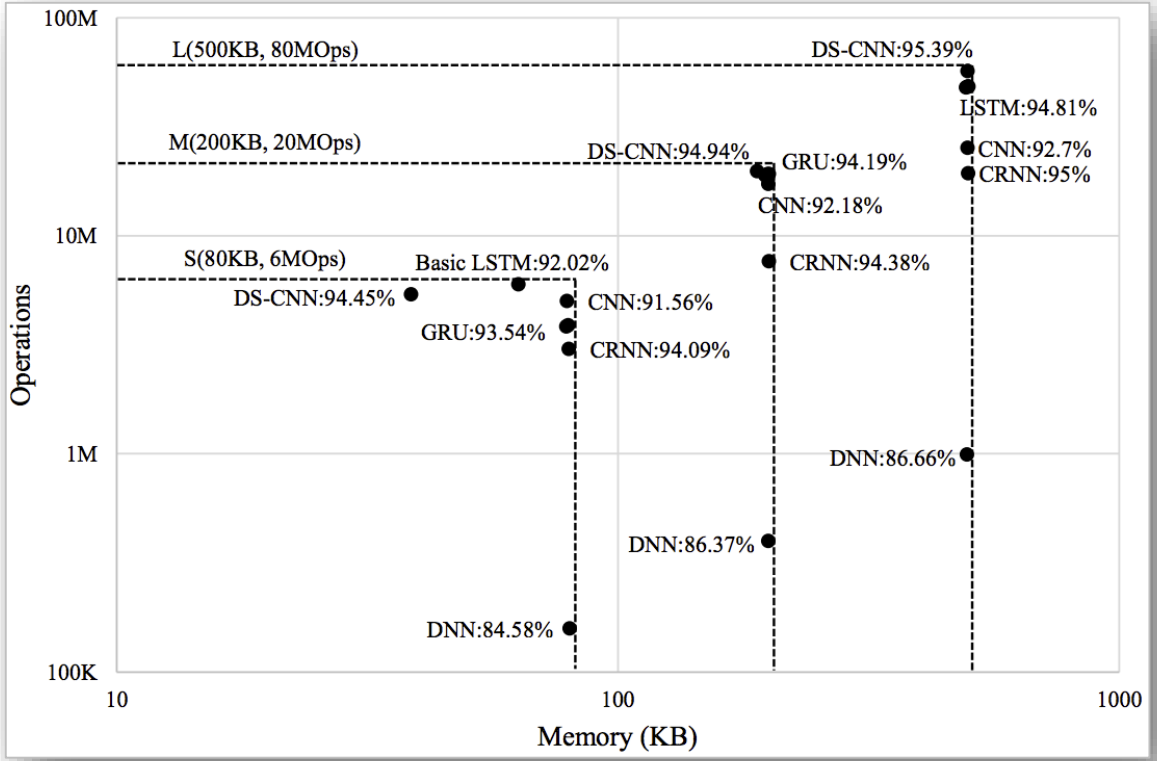
Classification

Neural network based – DNN, CNN, RNN (LSTM/GRU) or a mix of them

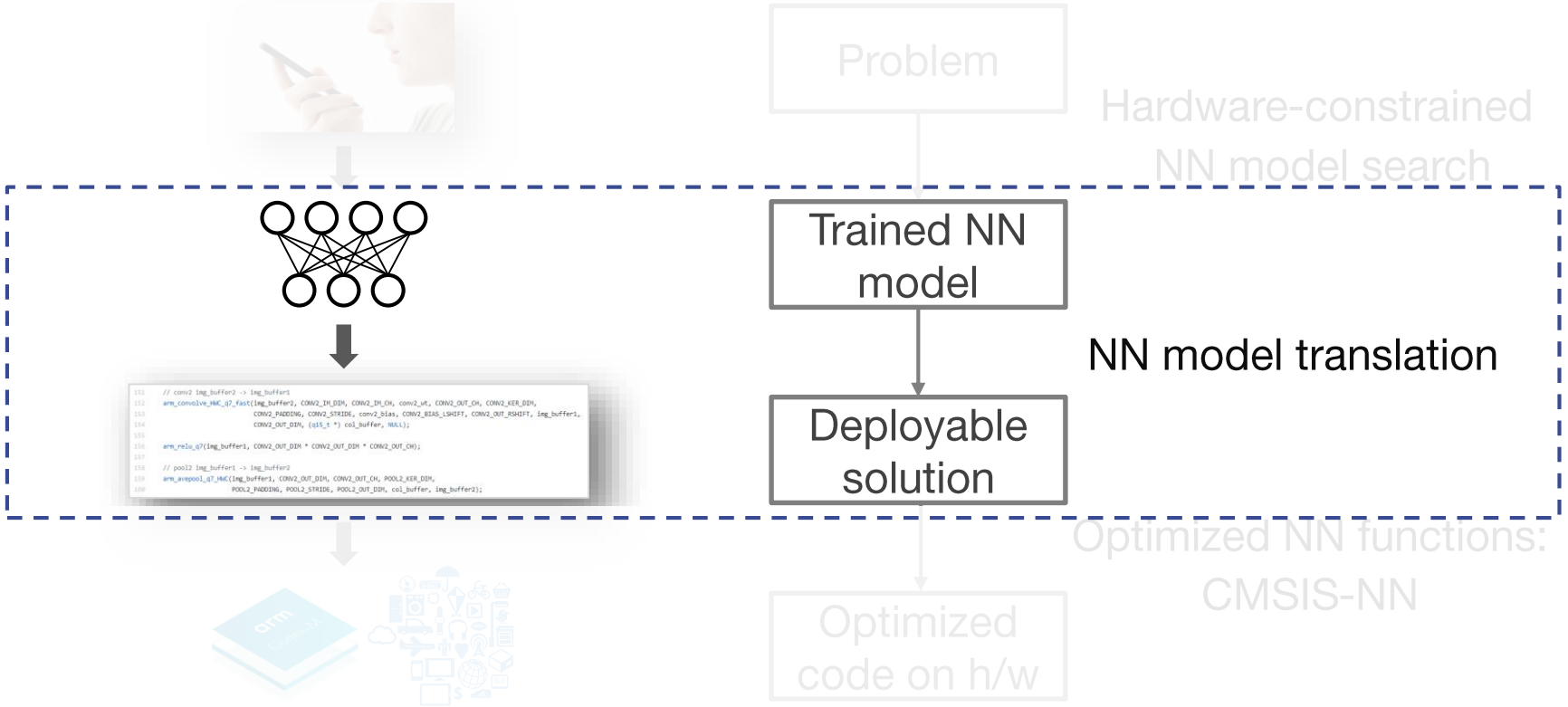
H/W Constrained NN Model Search



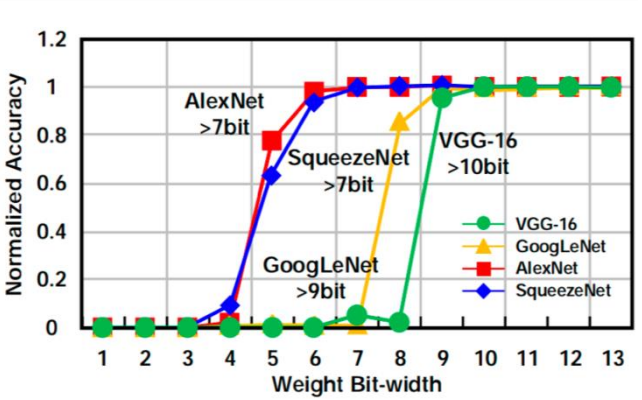
Optimize Models for the Platform



Developing NN Solutions on Cortex-M



NN Model Quantization



Neural networks can tolerate 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

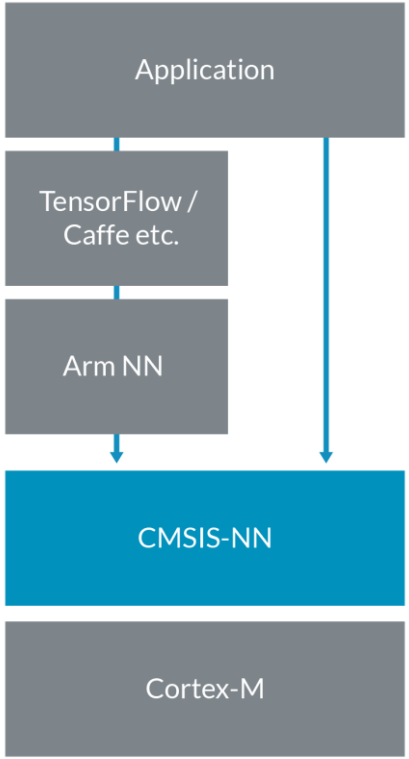
- Run quantization sweeps to identify optimal quantization strategy
- Quantize weights: does not need a dataset
- Quantize activations: run the model with dataset to extract ranges

Minimal loss in accuracy with 8-bit quantization (~0.1%)

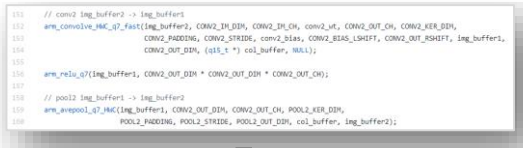
- Re-training can recover the accuracy loss in most cases
- May increase accuracy in some cases
- Regularization (or reduced over-fitting)

Model Deployment on Cortex-M MCUs

- Running ML framework on Cortex-M systems is impractical
- Need to run bare-metal code to efficiently use the limited resources
- Arm NN translates trained model to the code that runs on Cortex-M cores using CMSIS-NN functions
- **CMSIS-NN:** optimized low-level NN functions for Cortex-M CPUs
- CMSIS-NN APIs may also be directly used in the application code



Developing NN Solutions on Cortex-M



Problem

Trained NN model

Deployable solution

Optimized code on h/w

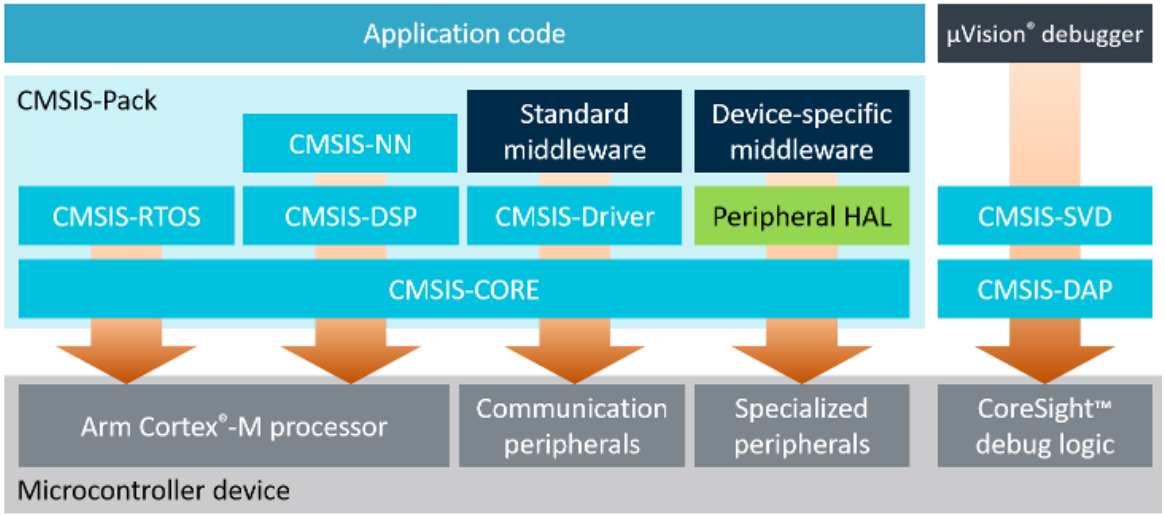
Hardware-constrained NN model search

NN model translation

Optimized NN functions: CMSIS-NN

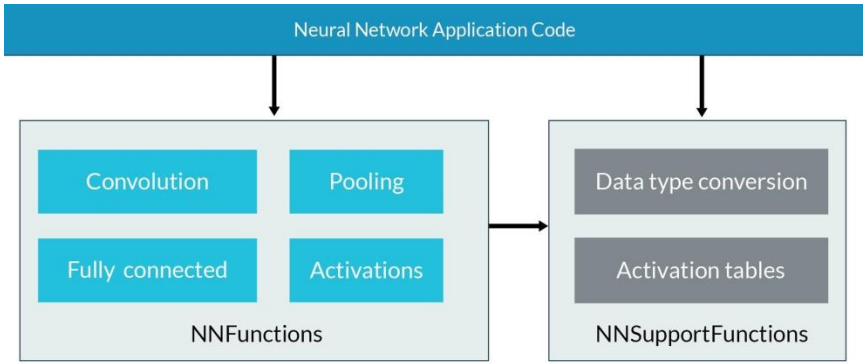
Cortex Microcontroller Software Interface Standard (CMSIS)

- 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/



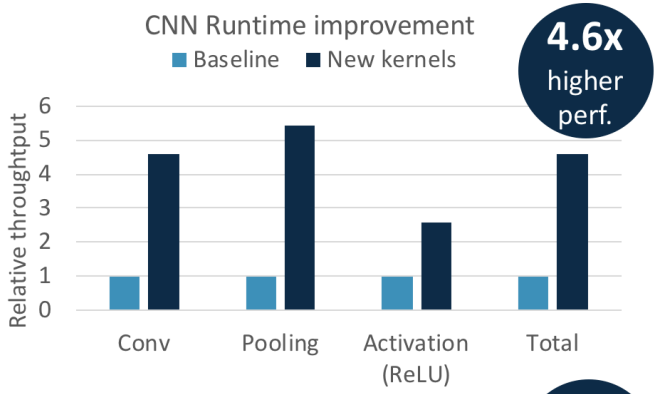
CMSIS-NN

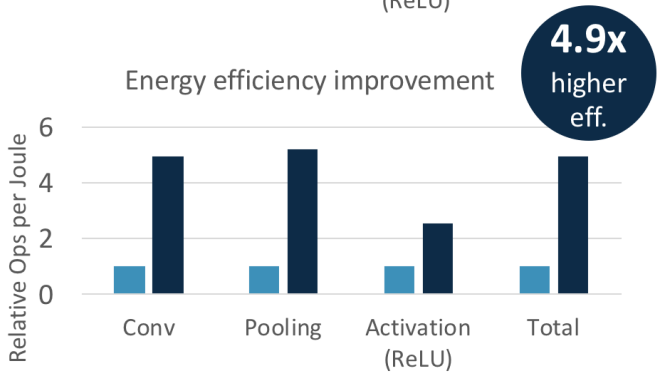
- CMSIS-NN: collection of optimized neural network functions for Cortex-M CPUs
- Key considerations:
  - Improve performance using SIMD instructions
  - Minimize memory footprint
  - NN-specific optimizations: data-layout and offline weight reordering



CMSIS-NN – Efficient NN kernels for Cortex-M CPUs

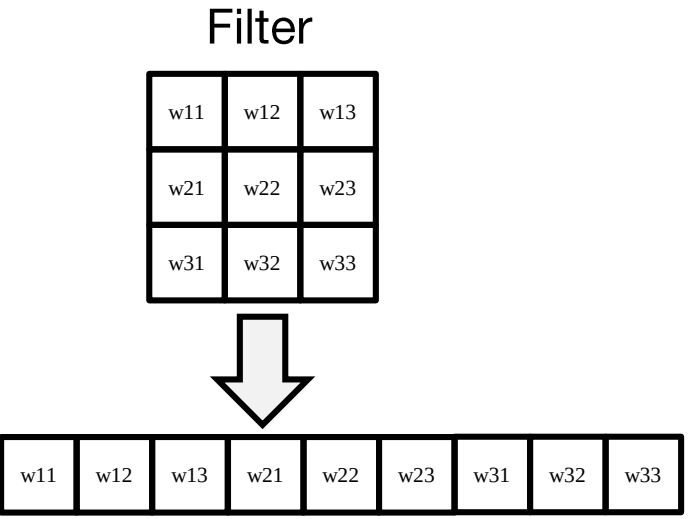
- **Convolution**
  - Boost compute density with GEMM based implementation
  - Reduce data movement overhead with depth-first data layout
  - Interleave data movement and compute to minimize memory footprint
- **Pooling**
  - Improve performance by splitting pooling into x-y directions
  - Improve memory access and footprint with in-situ updates
- **Activation**
  - ReLU: Improve parallelism by branch-free implementation
  - Sigmoid/Tanh: fast table-lookup instead of exponent computation

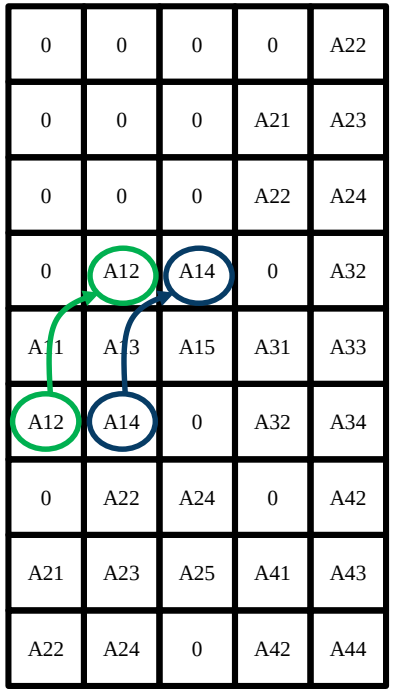


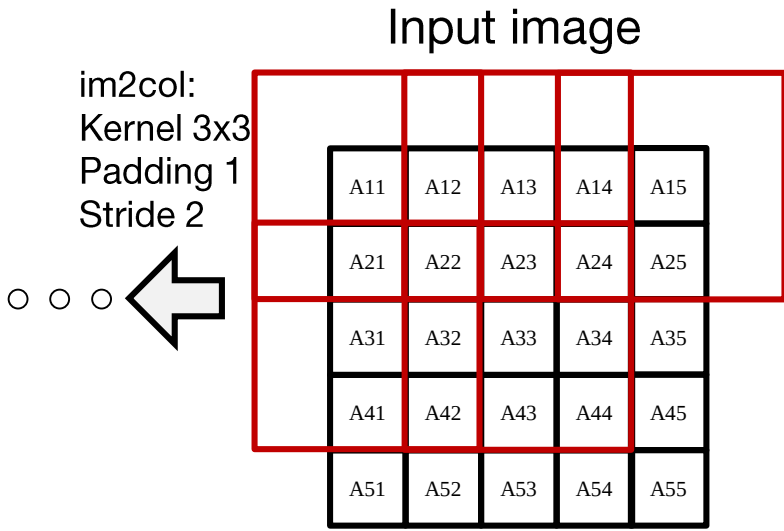


*Baseline uses CMSIS 1D Conv and Caffe-like Pooling/ReLU

Convolution: Im2col + Matrix Multiplication





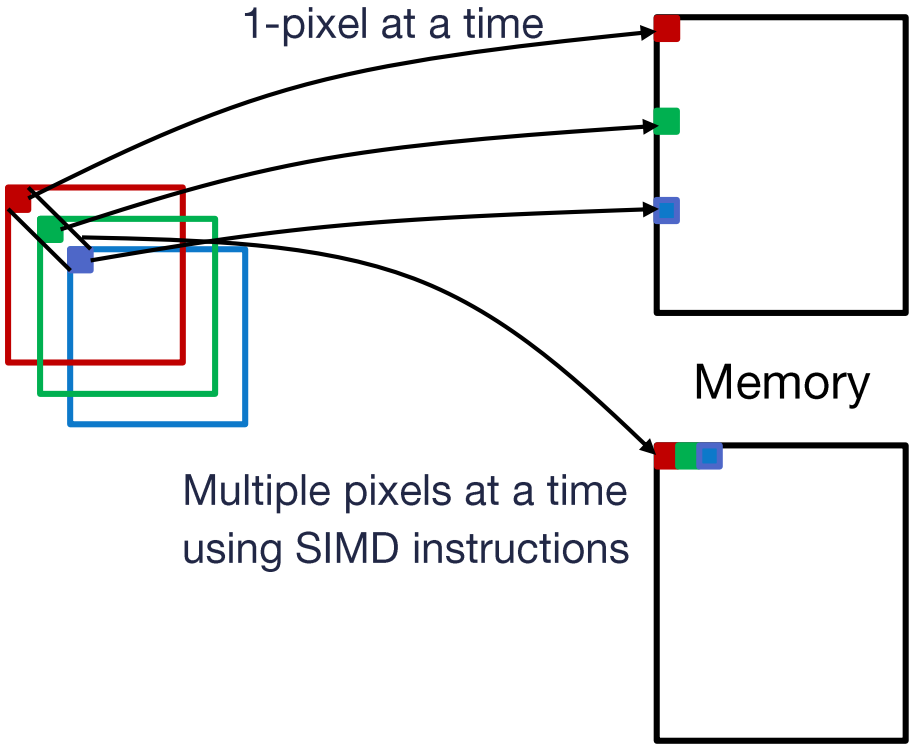


Solution: Partial im2col

- Generate 2 columns at a time
- Small run-time memory overhead

Memory foot-print increase: >10X

Data Layout in Memory: CHW vs. HWC

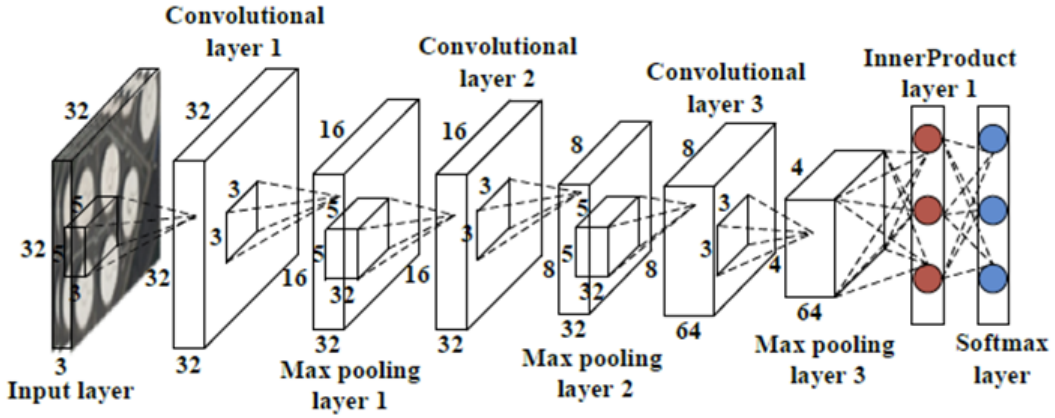


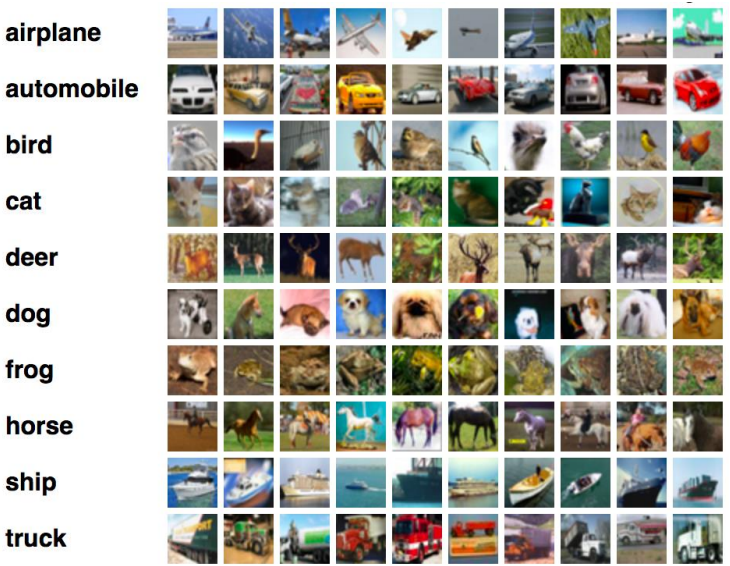
CHW
channel-last data layout
Inefficient data access

HWC
channel-first data layout
Efficient data access

Convolutional Neural Network (CNN) on Cortex-M7

- CIFAR-10 classification – classify images into 10 different object classes
- 3 convolution layers, 3 pooling layers and 1 fully-connected layer (~82% accuracy)





Convolutional Neural Network (CNN) on Cortex-M7

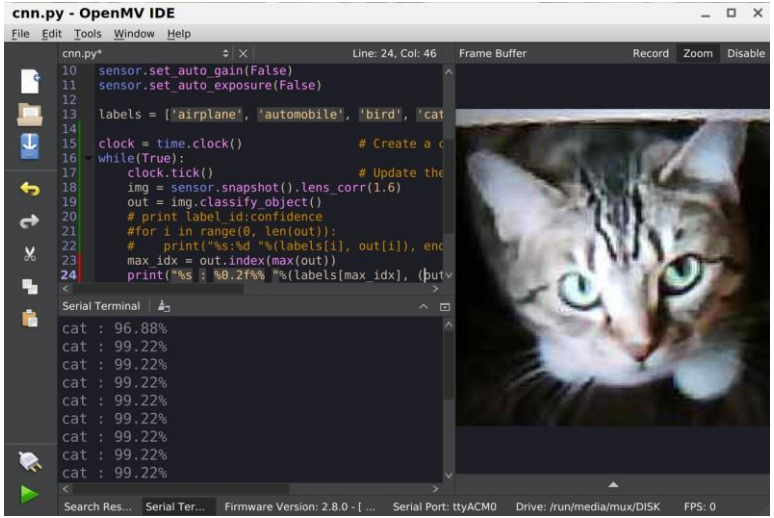
- CNN with 8-bit weights and 8-bit activations
- Total memory footprint: 87 kB weights + 40 kB activations + 10 kB buffers (I/O etc.)
- Example code available in CMSIS-NN github.

NUCLEO-F746ZG
216 MHz, 320 KB SRAM

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

Convolutional Neural Network (CNN) on Cortex-M7

OpenMV Cam with a Cortex-M7



Video :
https://www.youtube.com/watch?v=PdWi_fvY9Og

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: 66 KB weights + 1 KB activations + 2 KB buffers

Summary

- ML is moving to the IoT Edge
- Many ML use cases can be easily achieved with Cortex-M CPUs
- Exploration of neural network architectures is key for deployment on hardware-constrained devices
- CMSIS-NN: optimized neural network functions help achieve higher performance and higher energy efficiency

Resources

- CMSIS-NN paper: https://arxiv.org/abs/1801.06601
- CMSIS-NN blog: https://community.arm.com/processors/b/blog/posts/new-neural-network-kernels-boost-efficiency-in-microcontrollers-by-5x
- CMSIS-NN Github link: https://github.com/ARM-software/CMSIS_5/

- KWS (Keyword Spotting) paper: https://arxiv.org/abs/1711.07128
- KWS blog: 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/

- ArmNN: https://developer.arm.com/products/processors/machine-learning/arm-nn
- ArmNN SDK blog: https://community.arm.com/tools/b/blog/posts/arm-nn-sdk

Thank you!