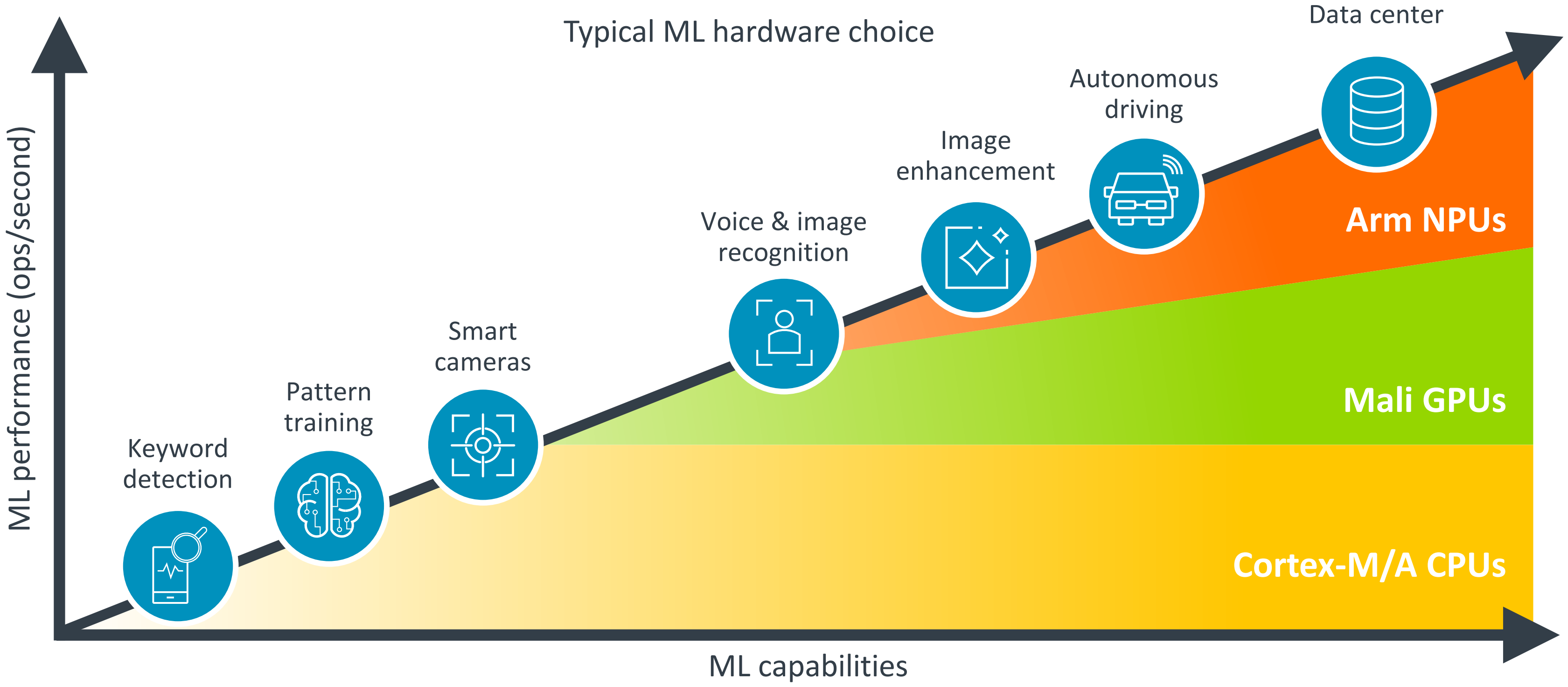
Machine Learning on Ultra-Constrained Devices

Mark Woods
Solutions Architect

Flexible, Scalable ML Solutions

Only Arm can enable ML everywhere



Deliver use cases with multiple hardware solutions

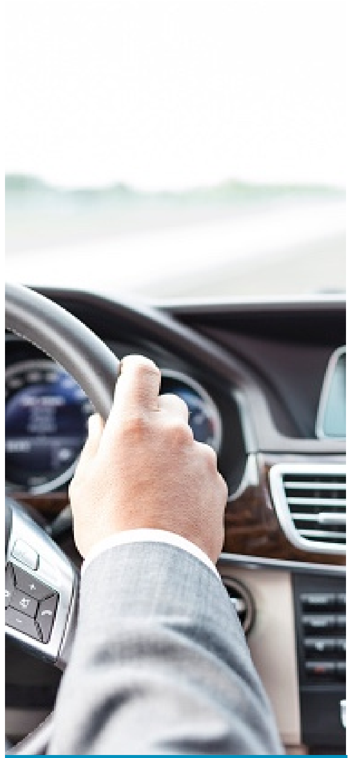
Choose best balance of ML performance versus capabilities per use case

Why ML is Moving to the Edge

Bandwidth

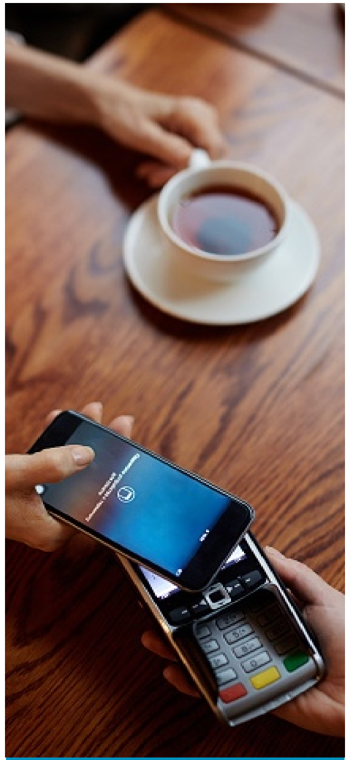
Power

Cost



Latency

Reliability



Security

Machine Learning and AI Basics

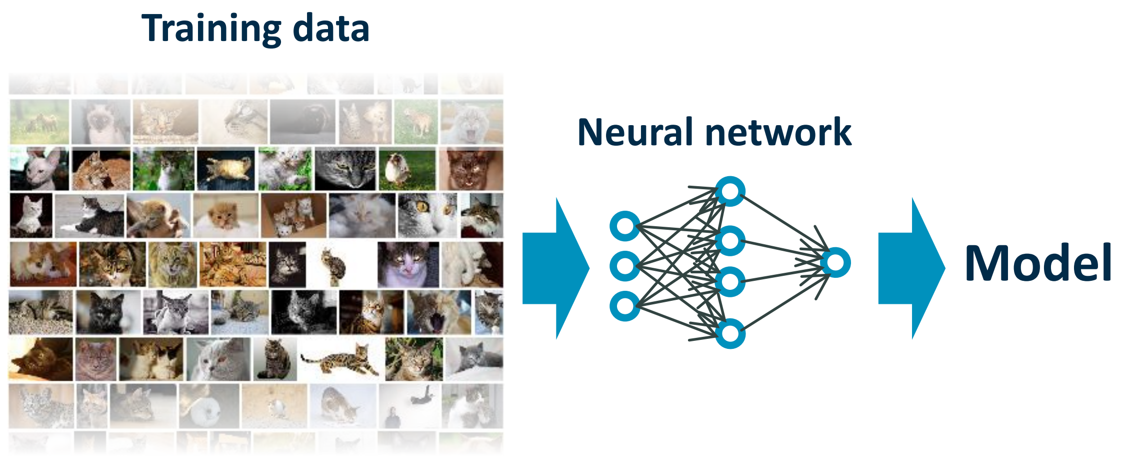
Artificial Intelligence: Umbrella term for machines acting like they are 'thinking'

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

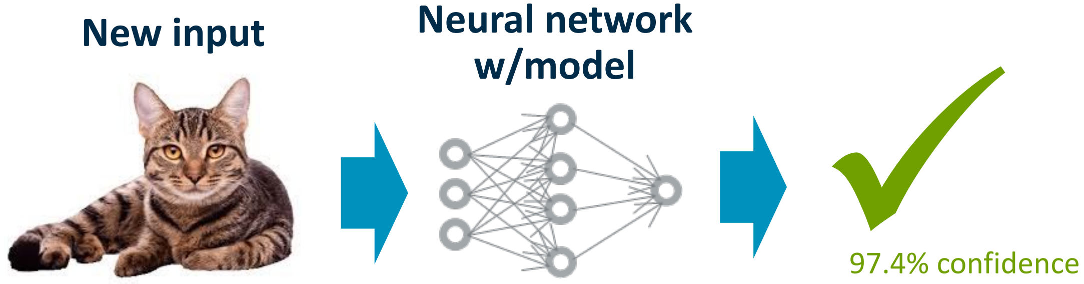
Deep Learning: machine learning using Deep Neural Network approaches

Algorithms: CNNs, RNNs, etc.

Training



Inference



Project Trillium: Arm's ML Computing Platform

Ecosystem

AI/ML Applications, Algorithms and Frameworks

TensorFlow™ | TensorFlow Lite™ | PyTorch™ | ONNX™

mxnet™ | Caffe™ | Caffe2™ | Android™ NNAPI

Software Products

Software Libraries Optimized for Arm Hardware

arm NN | arm COMPUTE LIBRARY | CMSIS-NN

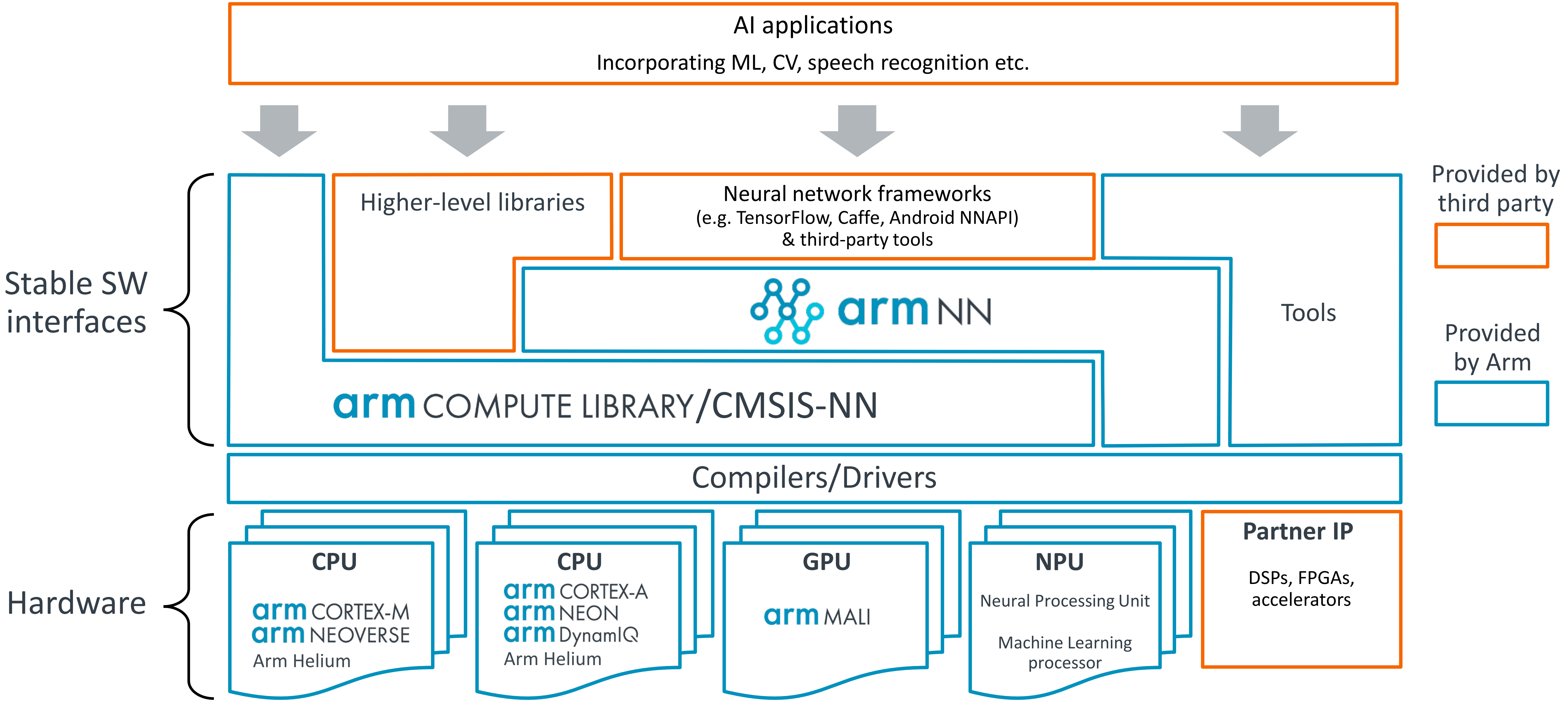
Hardware Products

Arm Hardware IP for AI/ML

CPU	GPU	NPU	Partner IP
arm CORTEX arm DynamIQ arm NEOVERSE	arm MALI	Machine Learning Processor (ML)	DSPs, FPGAs, Accelerators

arm

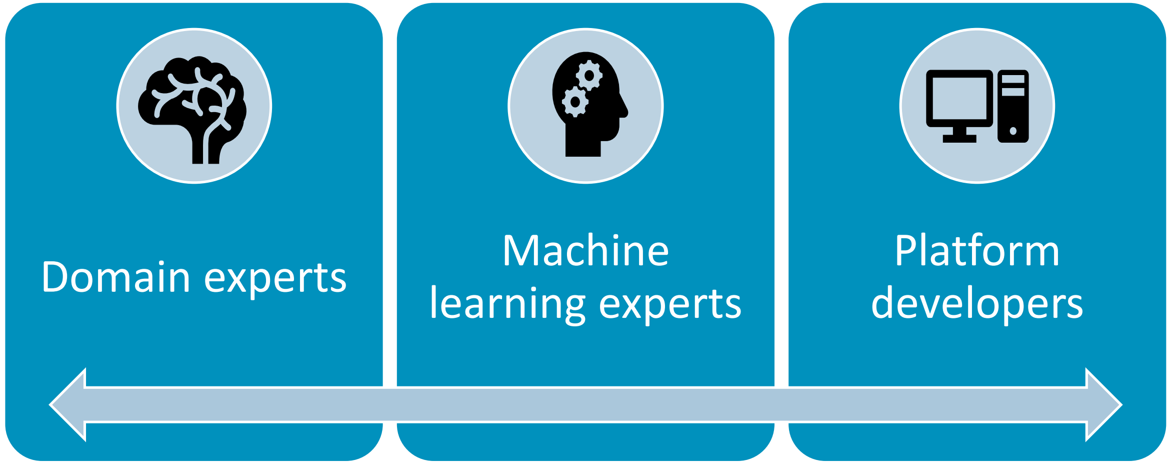
Arm's Heterogeneous Machine Learning Platform



Collaborate to Improve Standard ML Software Interface

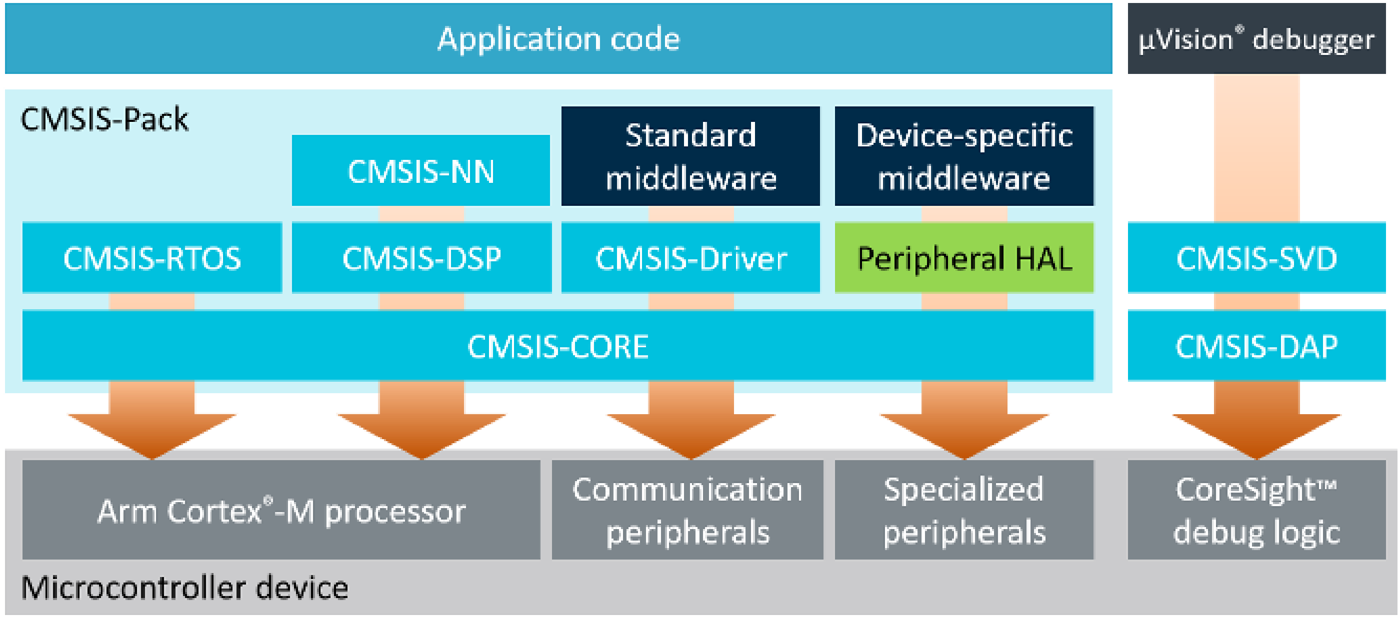
- Arm donated Arm NN to Linaro to accelerate development of a common software interface for ML
- Ways to help:
- Start contributing code
- Help us add exciting new features
- Join Linaro Machine Learning Initiative

- developer.arm.com/arm-nn



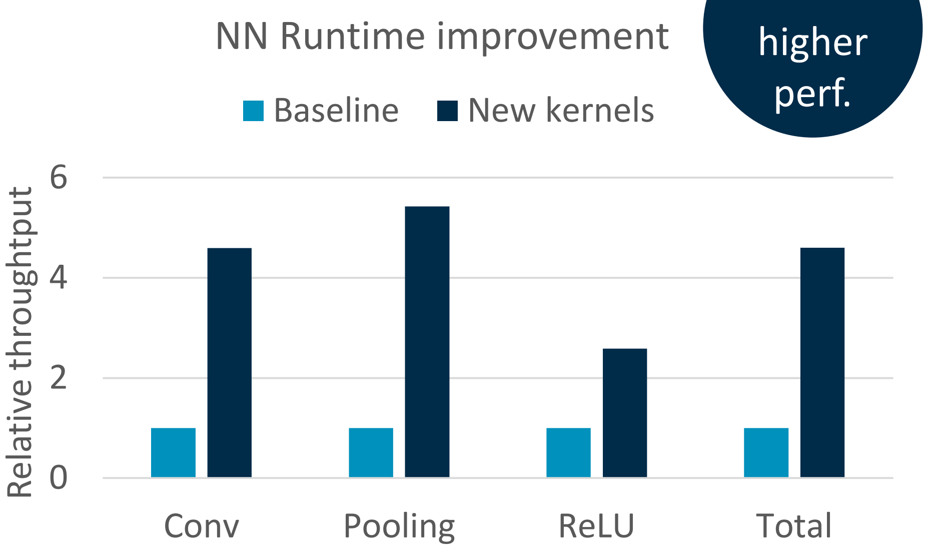
Cortex Microcontroller Software Interface Standard (CMSIS)

- 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/



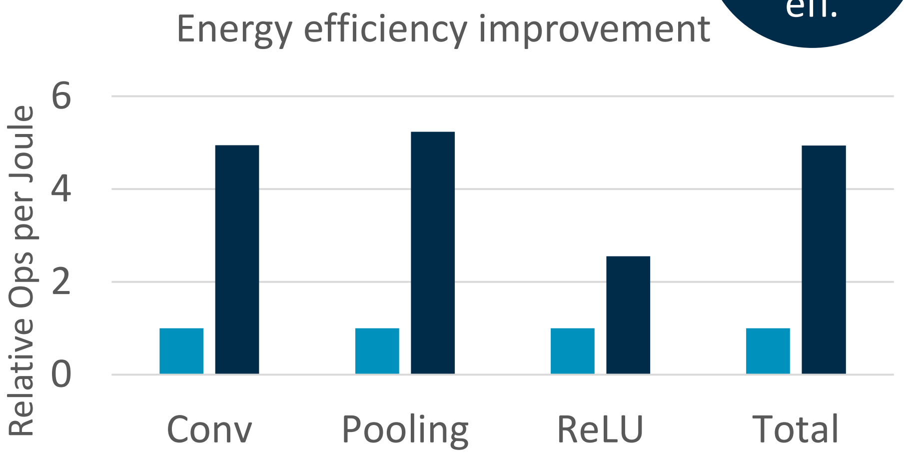
CMSIS-NN

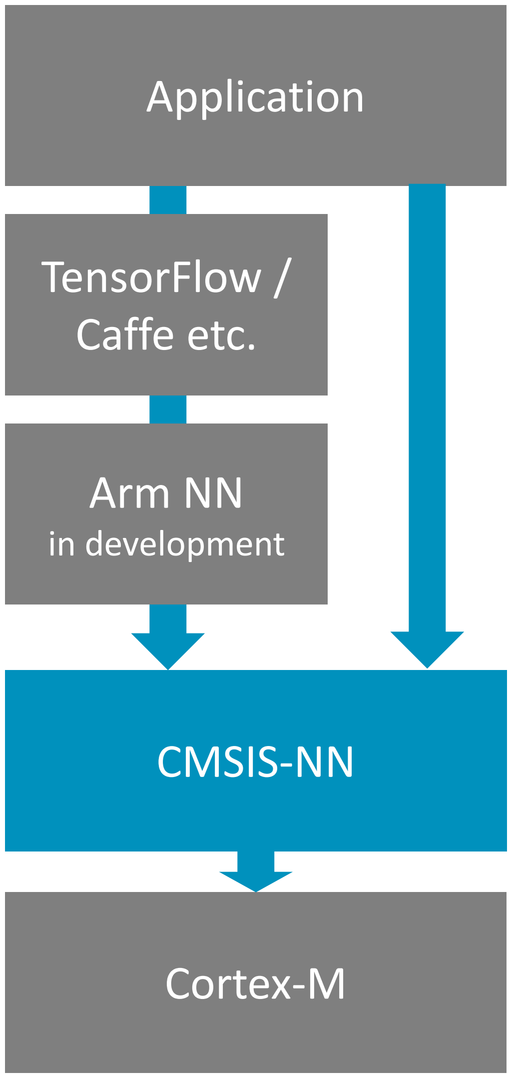
- Optimized low-level NN functions for Cortex-M CPUs
- A collection of efficient neural network kernels developed to maximize the performance and minimize the memory footprint of neural networks on Cortex-M processor cores
- Publicly available now (no fee, Apache 2.0 license)
  https://developer.arm.com/embedded/cmsis



4.6x higher perf.

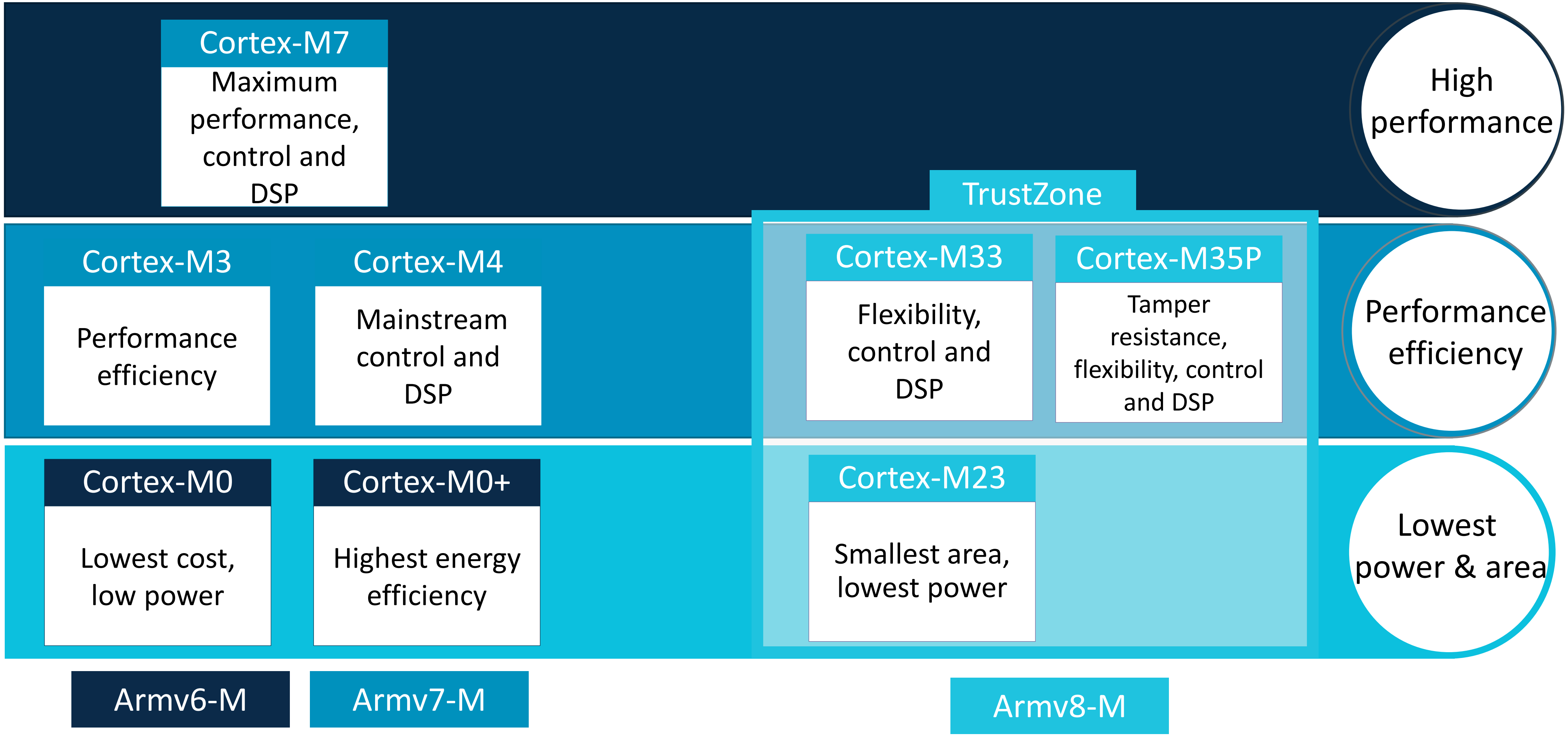
4.9x higher eff.



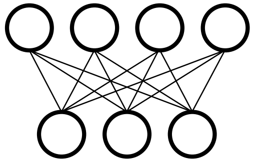


arm

Arm Cortex-M processor portfolio



Developing NN Solutions on Cortex-M



```
// conv2 img_buffer2 -> img_buffer1
arm_convolve_HWC_q7_fast(img_buffer2, CONV2_IM_DIM, CONV2_IM_CH, conv2_wt, CONV2_OUT_CH, CONV2_KER_DIM,
                         CONV2_PADDING, CONV2_STRIDE, conv2_bias, CONV2_BIAS_LSHIFT, CONV2_OUT_RSHIFT, img_buffer1,
                         CONV2_OUT_DIM, (q15_t *) col_buffer, NULL);

arm_relu_q7(img_buffer1, CONV2_OUT_DIM * CONV2_OUT_DIM * CONV2_OUT_CH);

// pool2 img_buffer1 -> img_buffer2
arm_avepool_q7_HWC(img_buffer1, CONV2_OUT_DIM, CONV2_OUT_CH, POOL2_KER_DIM,
                   POOL2_PADDING, POOL2_STRIDE, POOL2_OUT_DIM, col_buffer, img_buffer2);
```

Problem

↓ Hardware-constrained NN model search

Trained NN model

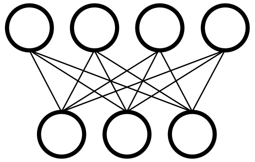
↓ NN model translation

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

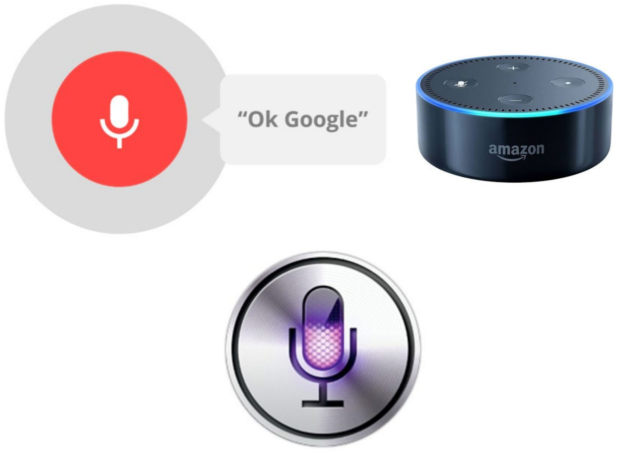
Deployable solution

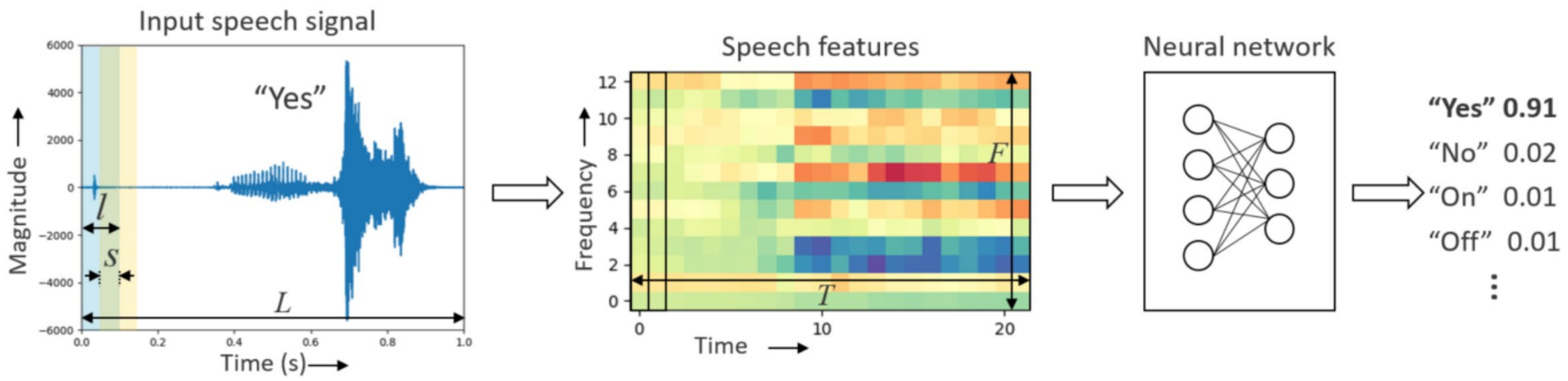
Optimized NN functions: CMSIS-NN

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

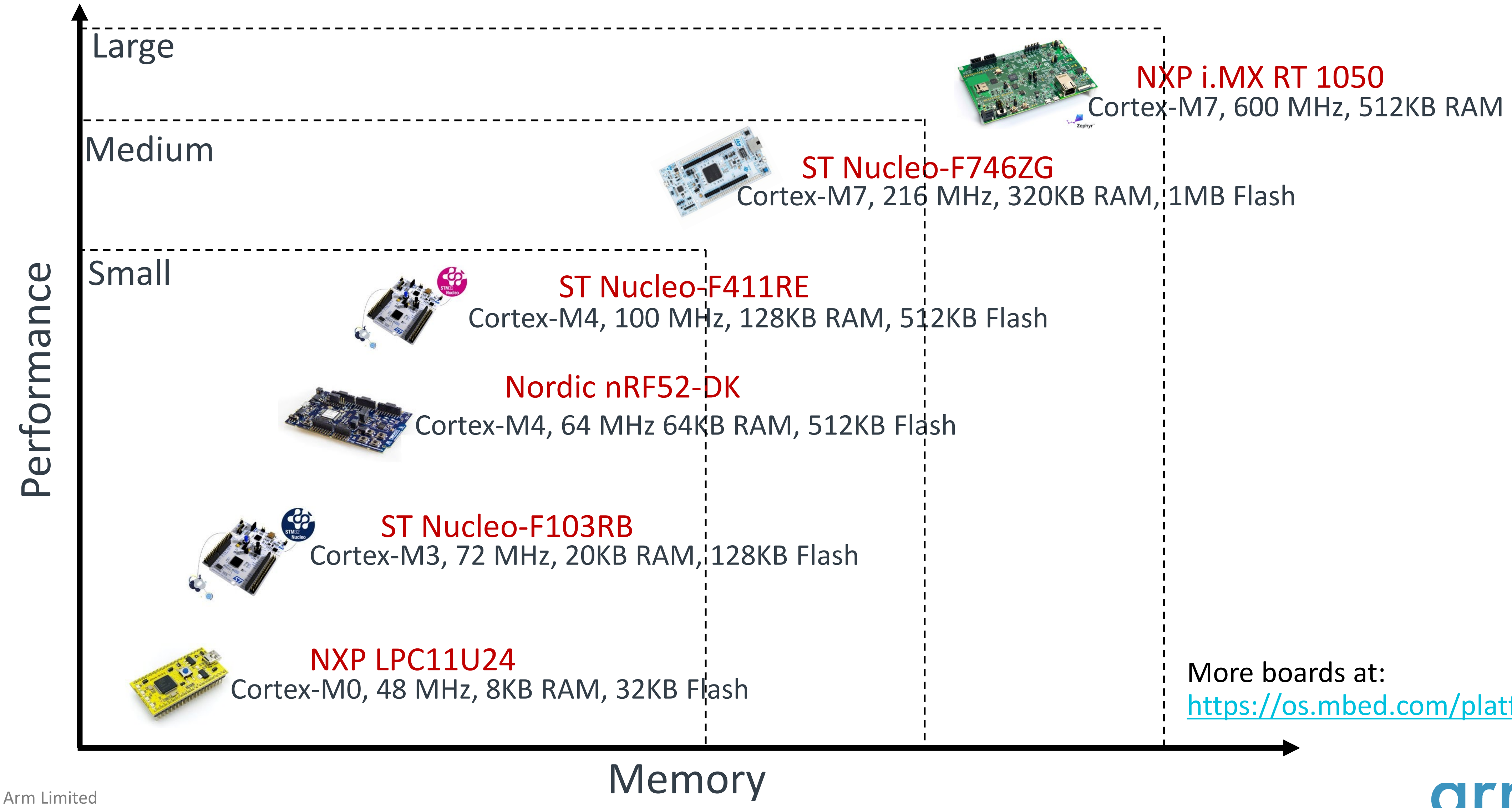
FFT-based mel frequency cepstral coefficients (MFCC) or log filter bank energies (LFBE)

Classification

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

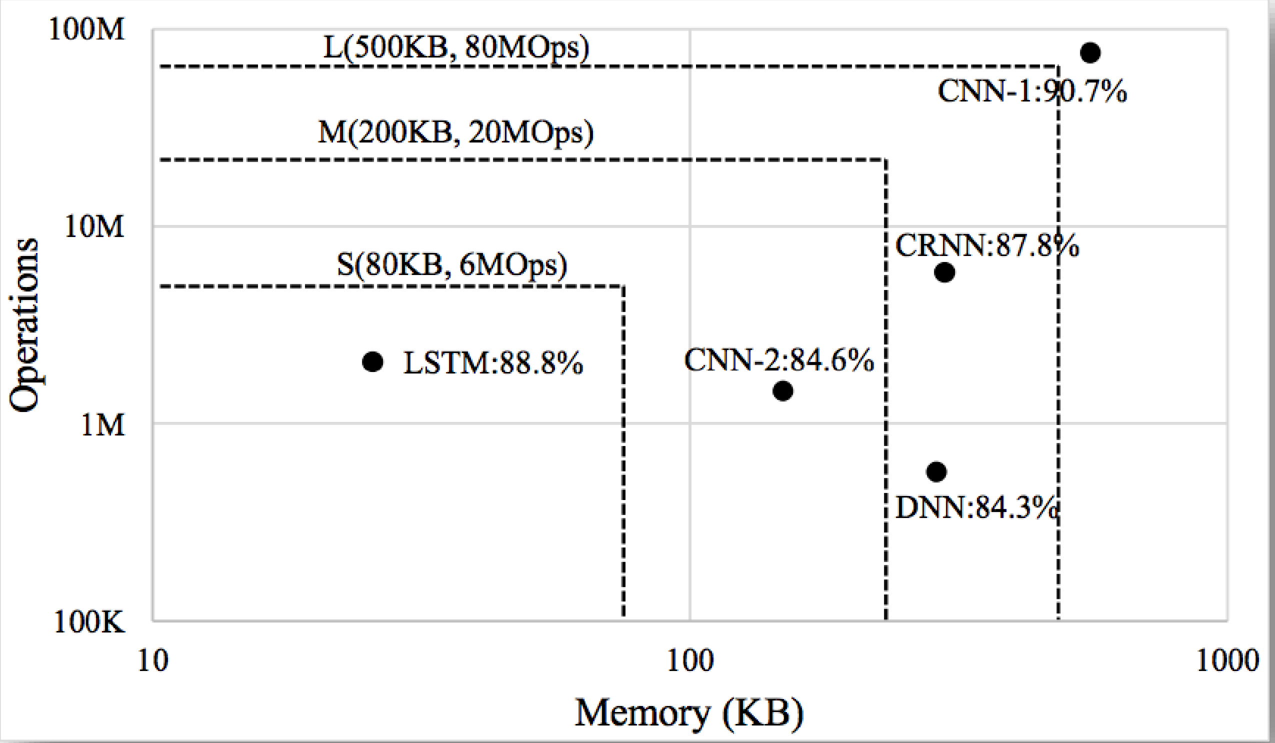
arm

Arm Cortex-M based MCU Platforms



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

Keyword Spotting on NN Models: Memory vs. Ops

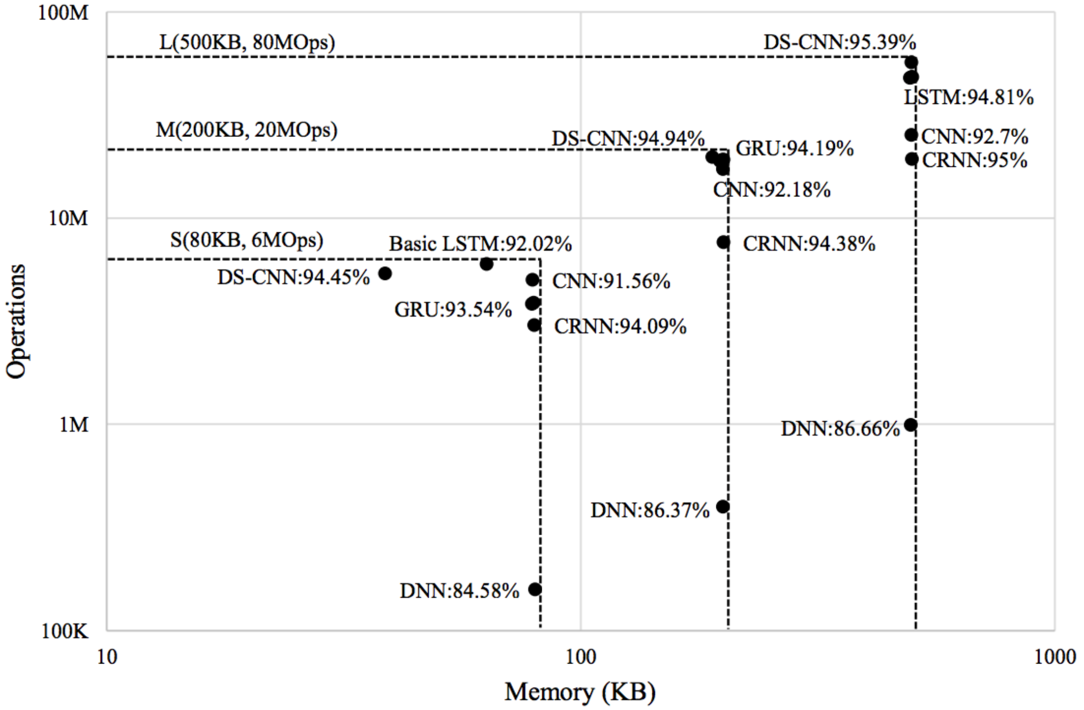


NN Models from literature trained on Google speech commands dataset

- 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

Summary of Best NN Models

NN model	S(80KB, 6MOps)			M(200KB, 20MOps)			L(500KB, 80MOps)		
	Acc.	Mem.	Ops	Acc.	Mem.	Ops	Acc.	Mem.	Ops
DNN	84.6%	80.0KB	158.8K	86.4%	199.4KB	397.0K	86.7%	496.6KB	990.2K
CNN	91.6%	79.0KB	5.0M	92.2%	199.4KB	17.3M	92.7%	497.8KB	25.3M
Basic LSTM	92.0%	63.3KB	5.9M	93.0%	196.5KB	18.9M	93.4%	494.5KB	47.9M
LSTM	92.9%	79.5KB	3.9M	93.9%	198.6KB	19.2M	94.8%	498.8KB	48.4M
GRU	93.5%	78.8KB	3.8M	94.2%	200.0KB	19.2M	94.7%	499.7KB	48.4M
CRNN	94.0%	79.7KB	3.0M	94.4%	199.8KB	7.6M	95.0%	499.5KB	19.3M
DS-CNN	94.4%	38.6KB	5.4M	94.9%	189.2KB	19.8M	95.4%	497.6KB	56.9M

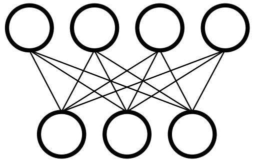


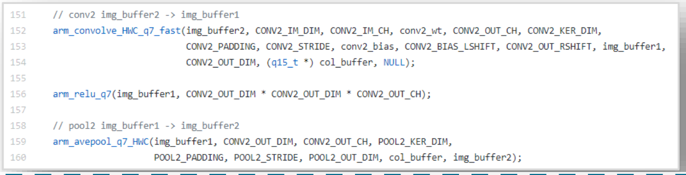
- Depthwise Separable CNN (DS-CNN) achieves highest accuracy
- Accuracy asymptotically reaches to 95%.

Hello Edge: Keyword spotting on Microcontrollers, arXiv: 1711.07128.

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

Developing NN Solutions on Cortex-M





```
// conv2 img_buffer2 -> img_buffer1
arm_convolve_HWC_q7_fast(img_buffer2, CONV2_IM_DIM, CONV2_IM_CH, conv2_wt, CONV2_OUT_CH, CONV2_KER_DIM,
                         CONV2_PADDING, CONV2_STRIDE, conv2_bias, CONV2_BIAS_LSHIFT, CONV2_OUT_RSHIFT, img_buffer1,
                         CONV2_OUT_DIM, (q15_t *) col_buffer, NULL);

arm_relu_q7(img_buffer1, CONV2_OUT_DIM * CONV2_OUT_DIM * CONV2_OUT_CH);

// pool2 img_buffer1 -> img_buffer2
arm_avepool_q7_HWC(img_buffer1, CONV2_OUT_DIM, CONV2_OUT_CH, POOL2_KER_DIM,
                   POOL2_PADDING, POOL2_STRIDE, POOL2_OUT_DIM, col_buffer, img_buffer2);
```


Problem

Hardware-constrained NN model search

Trained NN model

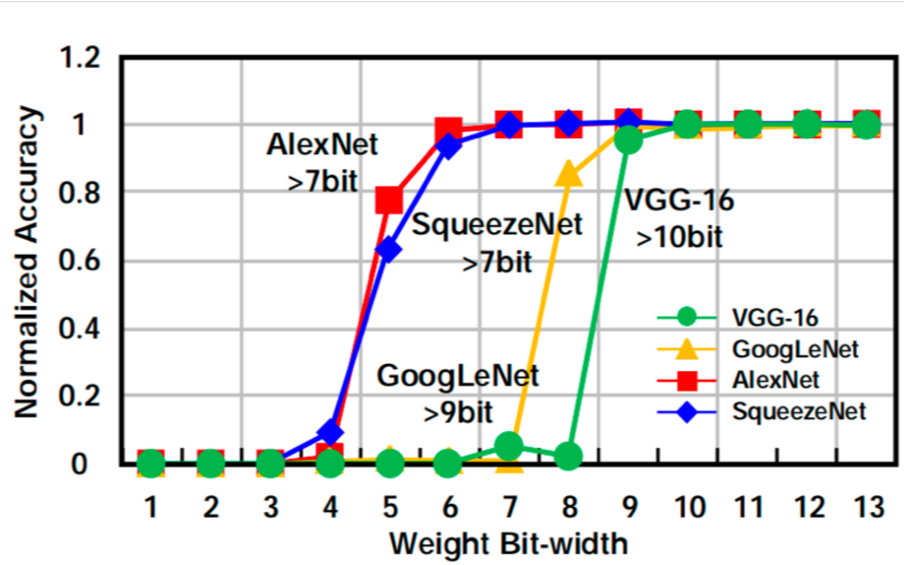
NN model translation

Deployable solution

Optimized NN functions: CMSIS-NN

Optimized code on h/w

NN Model Quantization



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

NN model	32-bit floating point model accuracy			8-bit quantized model accuracy		
	Train	Val.	Test	Train	Val.	Test
DNN	97.77%	88.04%	86.66%	97.99%	88.91%	87.60%
Basic LSTM	98.38%	92.69%	93.41%	98.21%	92.53%	93.51%
GRU	99.23%	93.92%	94.68%	99.21%	93.66%	94.68%
CRNN	98.34%	93.99%	95.00%	98.43%	94.08%	95.03%

- Minimal loss in accuracy (~0.1%).
- May increase accuracy in some cases
  - Regularization (or reduced over-fitting)

Developing NN Solutions on Cortex-M


```c
// conv2 img_buffer2 -> img_buffer1
arm_convolve_HWC_q7_fast(img_buffer2, CONV2_IM_DIM, CONV2_IM_CH, conv2_wt, CONV2_OUT_CH, CONV2_KER_DIM,
                         CONV2_PADDING, CONV2_STRIDE, conv2_bias, CONV2_BIAS_LSHIFT, CONV2_OUT_RSHIFT, img_buffer1,
                         CONV2_OUT_DIM, (q15_t *) col_buffer, NULL);

arm_relu_q7(img_buffer1, CONV2_OUT_DIM * CONV2_OUT_DIM * CONV2_OUT_CH);

// pool2 img_buffer1 -> img_buffer2
arm_avepool_q7_HWC(img_buffer1, CONV2_OUT_DIM, CONV2_OUT_CH, POOL2_KER_DIM,
                   POOL2_PADDING, POOL2_STRIDE, POOL2_OUT_DIM, col_buffer, img_buffer2);
```

Problem

→ Hardware-constrained NN model search

Trained NN model

→ NN model translation

Deployable solution

→ Optimized NN functions: CMSIS-NN

Optimized code on h/w

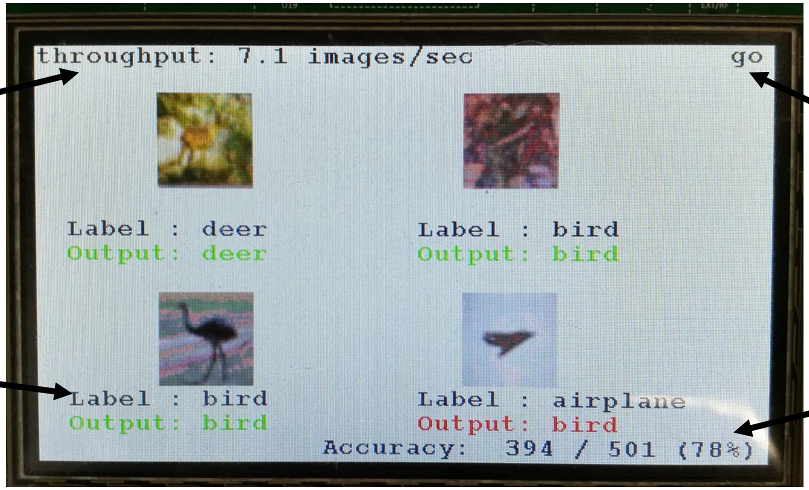
Demo - Multiple Neural Networks on Cortex-M7

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

CNN throughput

Image label and classification output

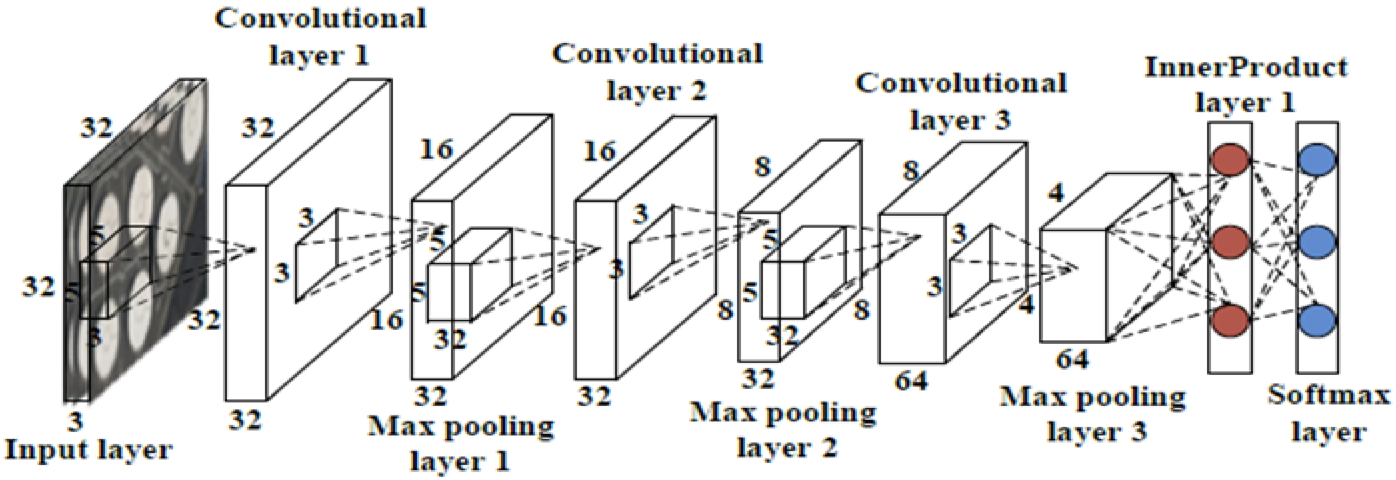


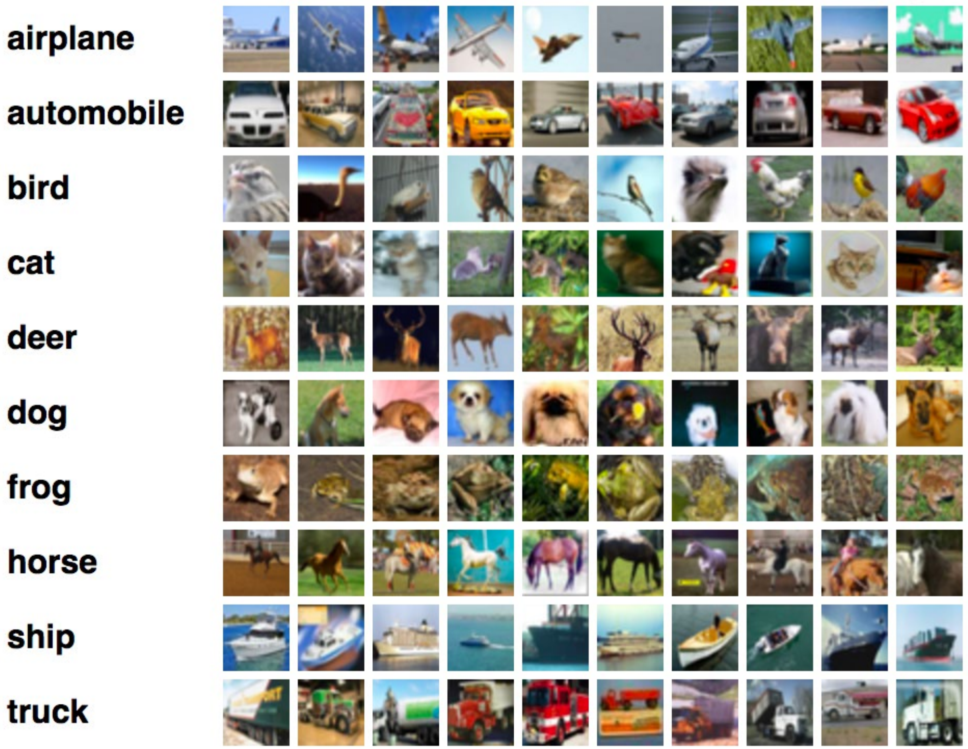
Detected voice command

Classification accuracy

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

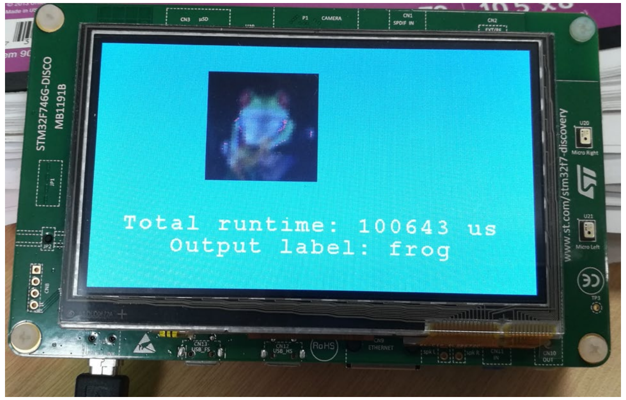




CIFAR-10 dataset: https://www.cs.toronto.edu/~kriz/cifar.html

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

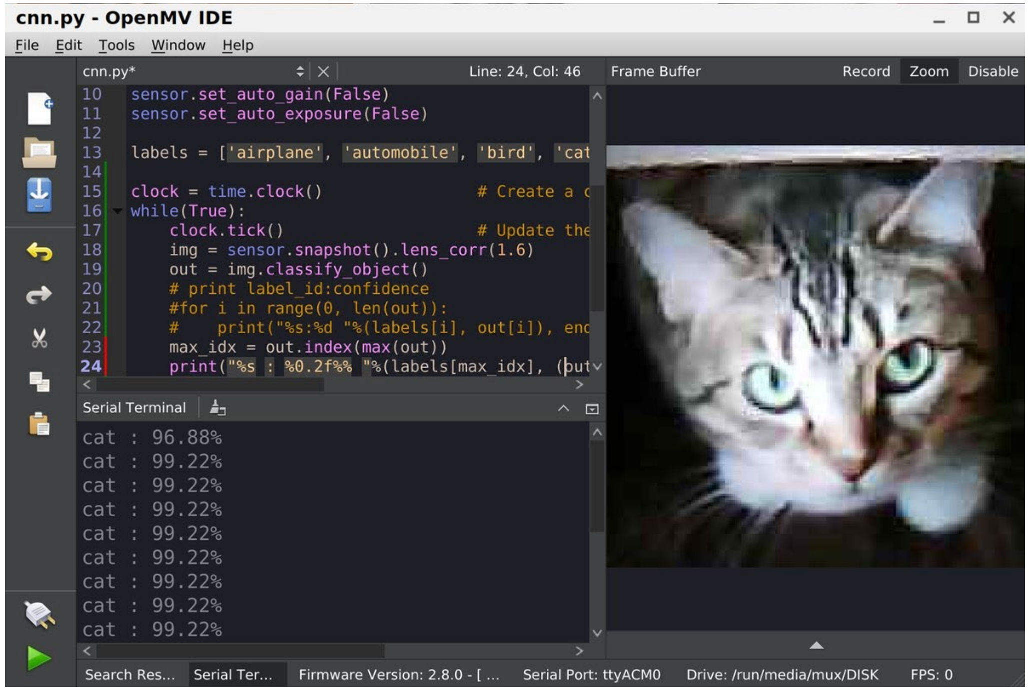


NUCLEO-F746ZG
216 MHz, 320 KB SRAM

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.io Cam with a Cortex-M7

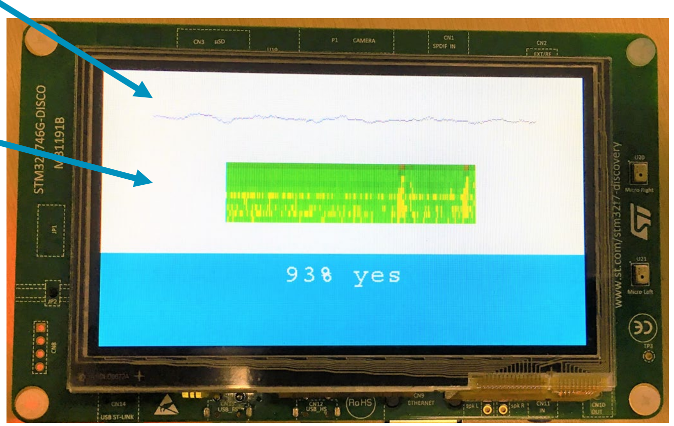


Video: https://www.youtube.com/watch?v=PdWi_fvY9Og

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

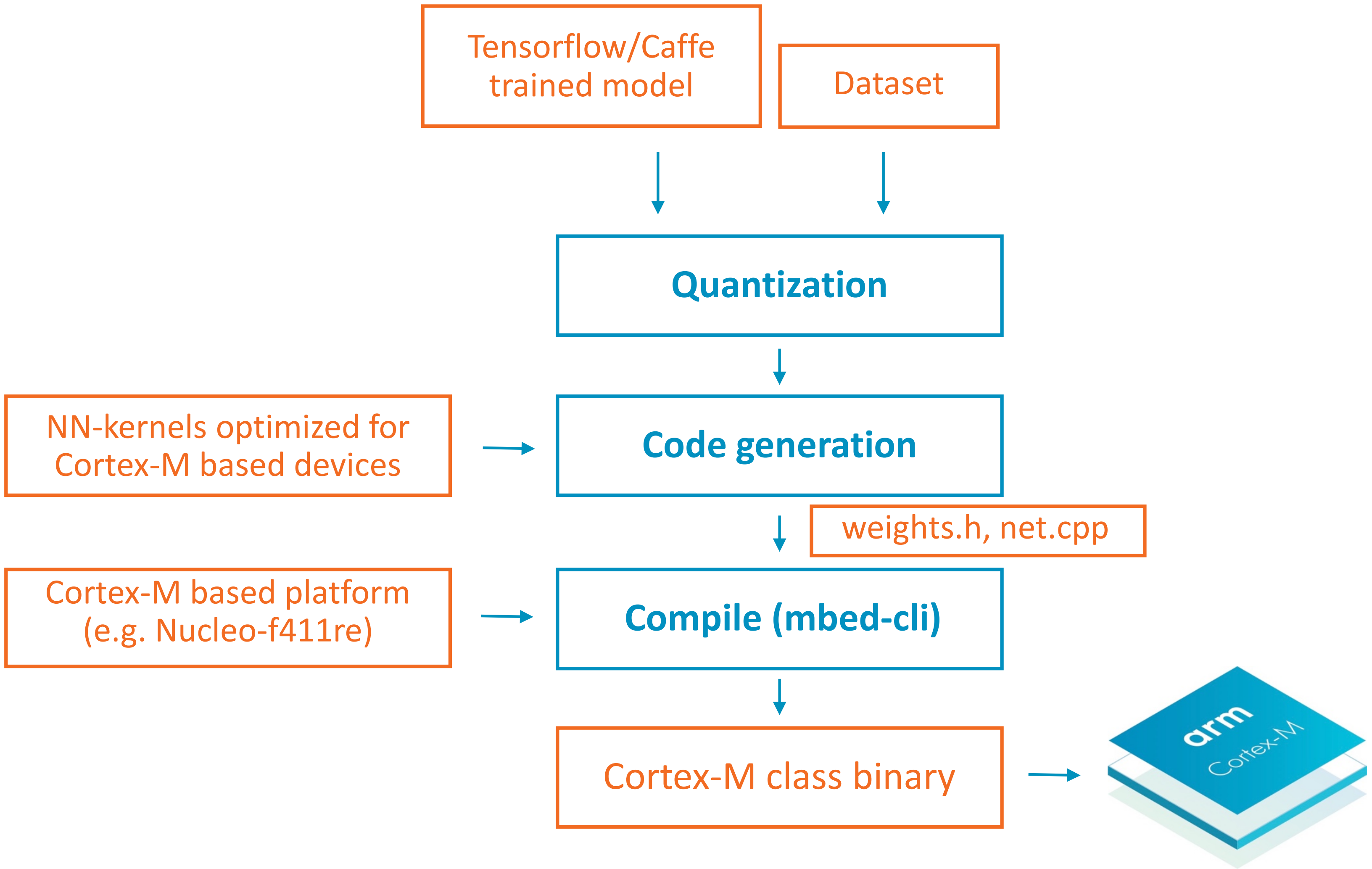


Demo with DNN, 8-bit weights and 8-bit activations

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

	Cortex-M4			Cortex-M7		
	Accuracy	Memory	Ops	Accuracy	Memory	Ops
DNN	85.4 %	67.0K	133.0K	85.4 %	98.0K	194.4K
Basic LSTM	91.9 %	43.1K	4.0M	92.5 %	79.3K	7.4M
LSTM	91.7%	54.5K	5.1M	92.4%	96.2K	9.1M
GRU	92.4%	60.1K	5.6M	92.4%	60.1K	5.6M
CNN-GRU	92.2%	66.8K	2.5M	94.1%	196.3K	7.8M
Depthwise CNN	92.2%	55.1K	5.9M	93.8%	178.5K	16.4M

Machine Learning Model Deployment on Arm Cortex-M



Get started today with a wealth of resources from Arm

- 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
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
تشكر