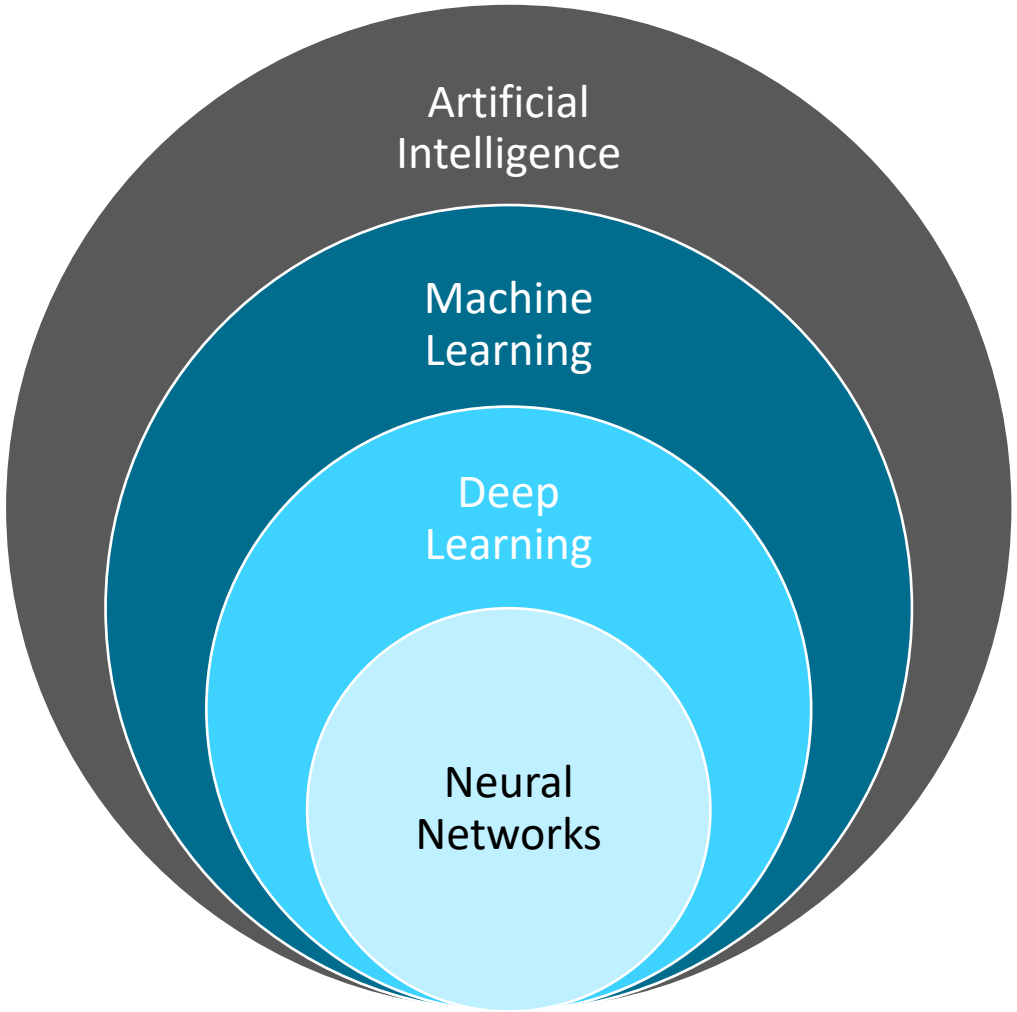
Deep Learning on Arm Cortex-M Microcontrollers

Rod Crawford

Director Software Technologies, Arm

What is Machine Learning (ML)?



Additional terms

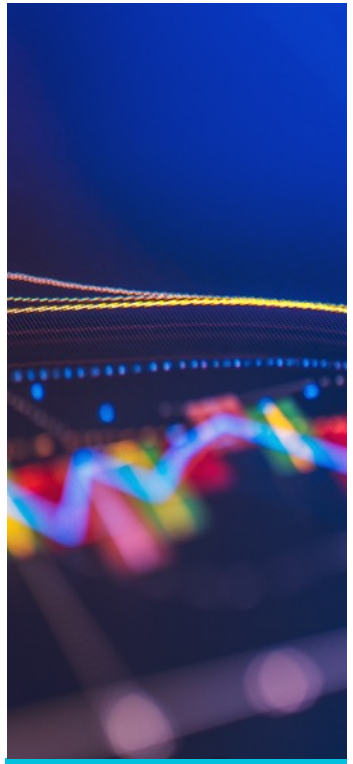
- **Location**
  - **Cloud** – processing done in data farms
  - **Edge** – processing done in local devices (growing much faster than Cloud ML)
- **Key components of machine learning**
  - **Model** – a mathematical approximation of a collection of input data
  - **Training** – in deep learning, data-sets are used to create a ‘model’
  - **Inference** – in deep learning, a ‘model’ is used to check against new data

© 2018 Arm Limited
arm

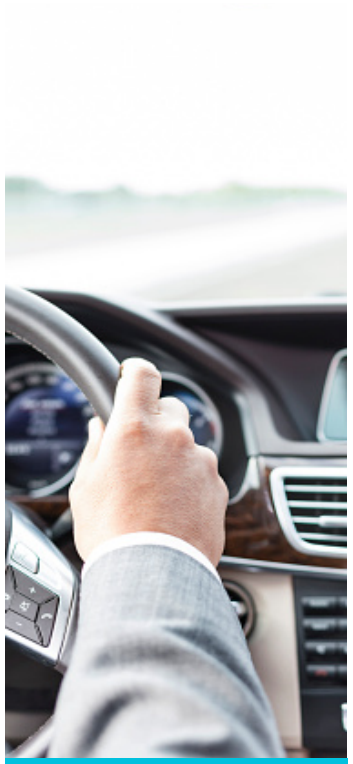
Why is ML Moving to the Edge

Bandwidth

Power

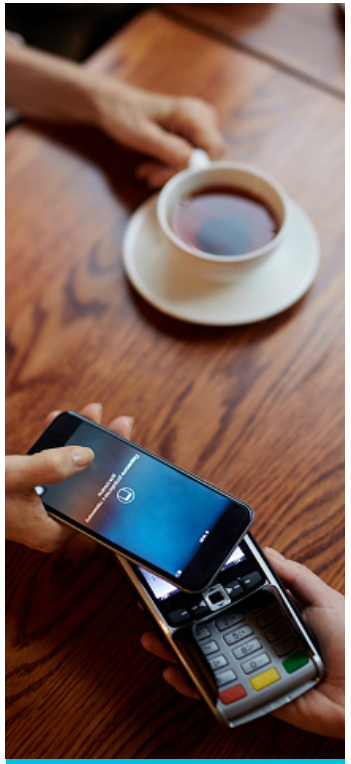


Cost



Latency

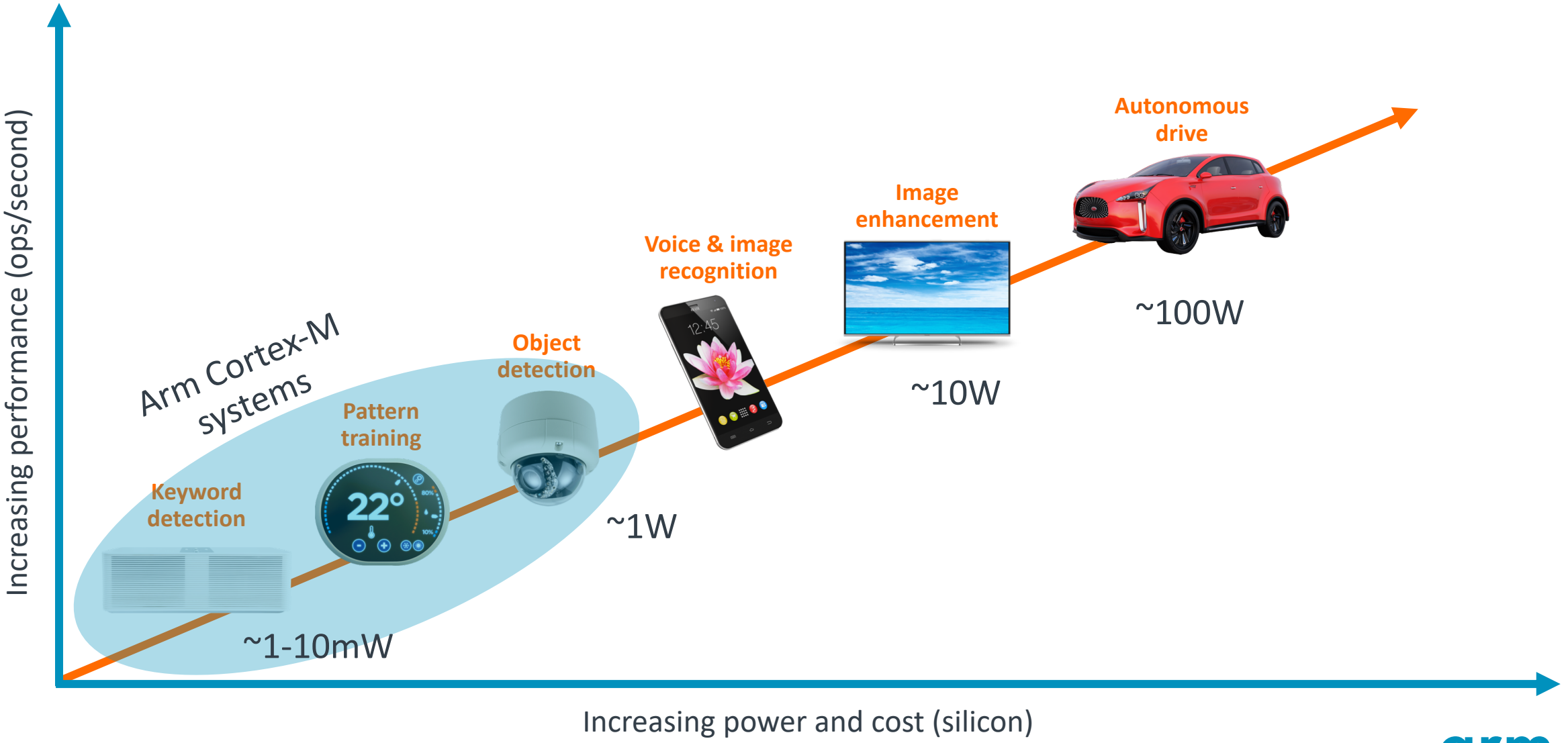
Reliability



Security

© 2018 Arm Limited
arm

ML Edge Use cases

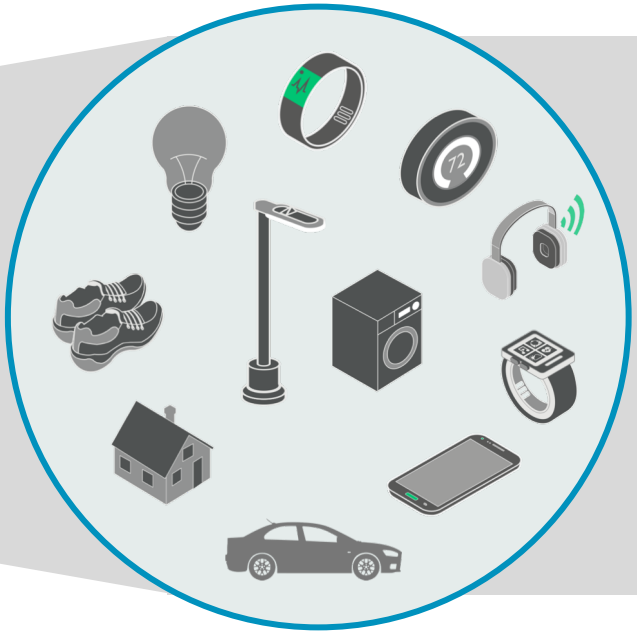


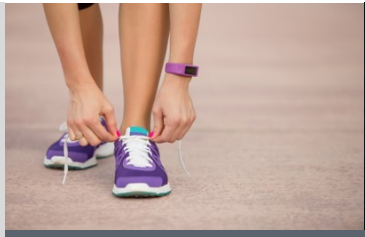
© 2018 Arm Limited

Use cases demand more embedded intelligence

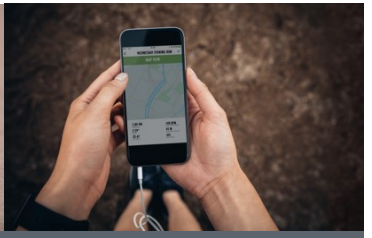
Arm Cortex-M

Expanding opportunity for the embedded intelligence market

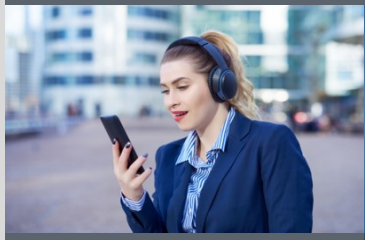




Sensor fusion



Contextual awareness



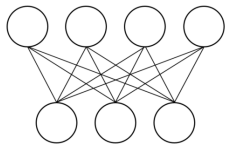
Voice activation

Biometrics

More data → More processing → More algorithms → More compute

© 2018 Arm Limited
arm

Developing NN Solution on Cortex-M MCUs



Problem

Trained NN model

Deployable Solution

Optimized code on h/w

Hardware-constrained NN model search

NN model translation

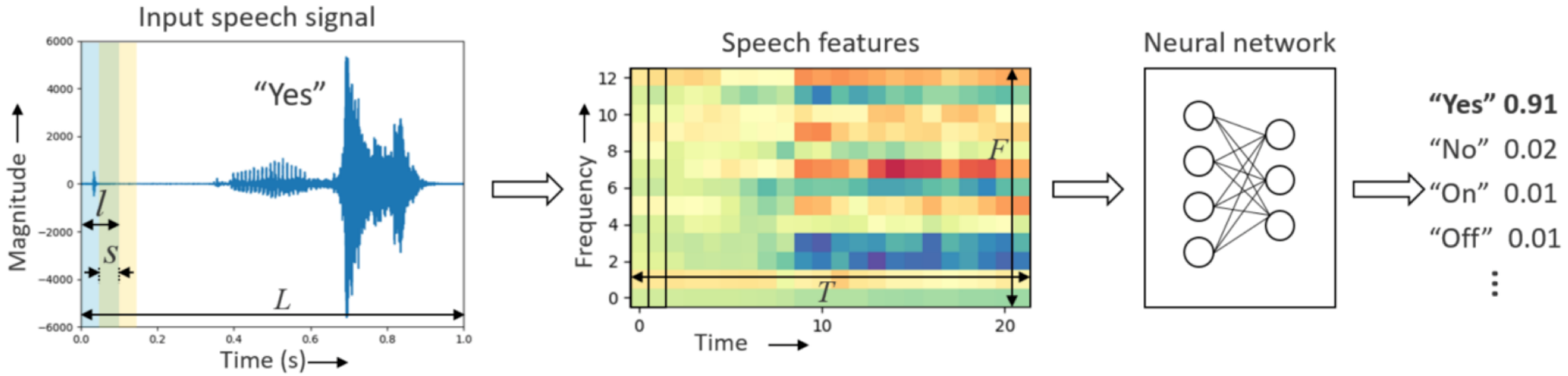
Optimized NN functions: CMSIS-NN

© 2018 Arm Limited
arm

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

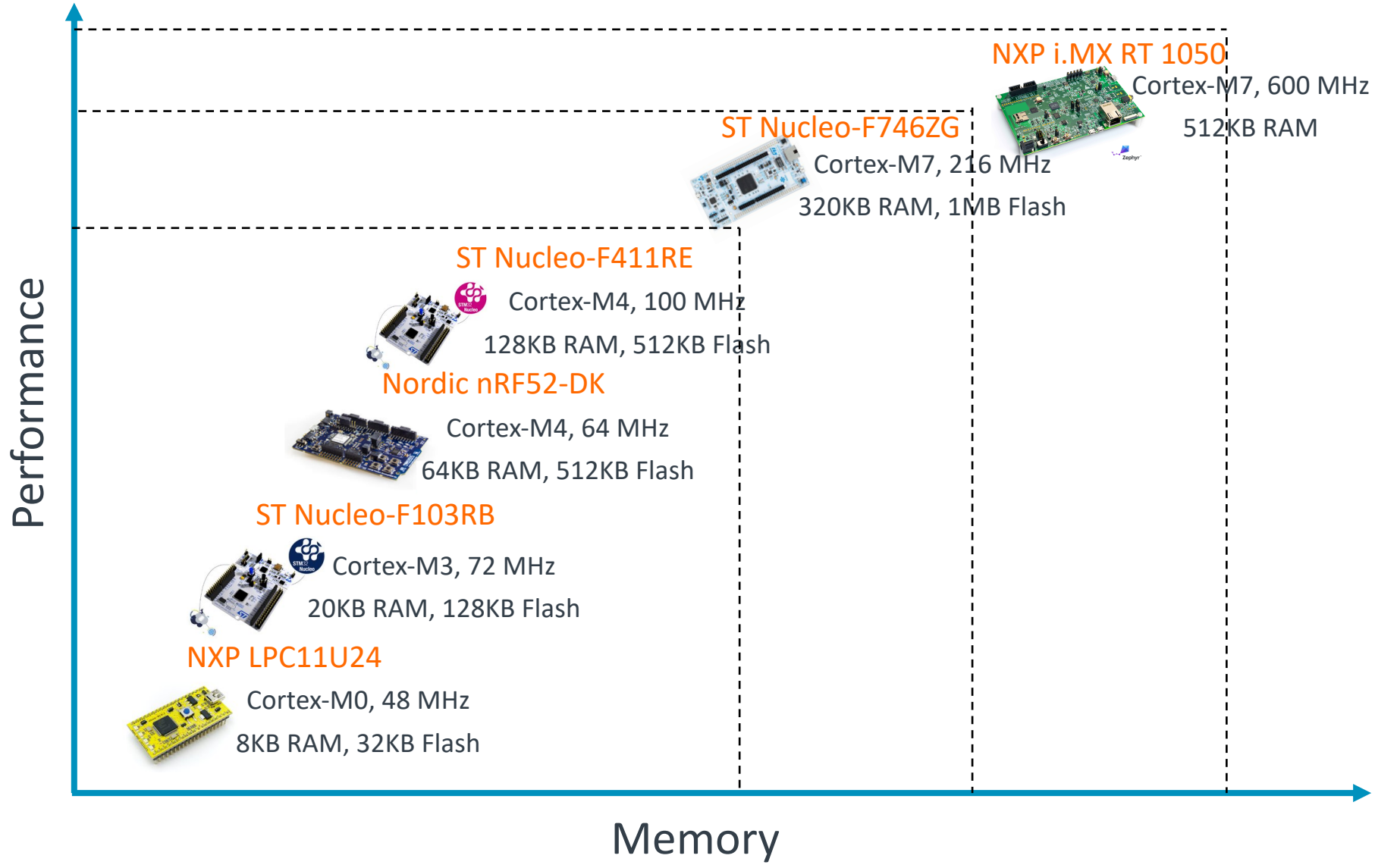
FFT-based mel frequency cepstral coefficients (MFCC) or log filter bank energies (LFBE).

Classification

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

© 2018 Arm Limited
arm

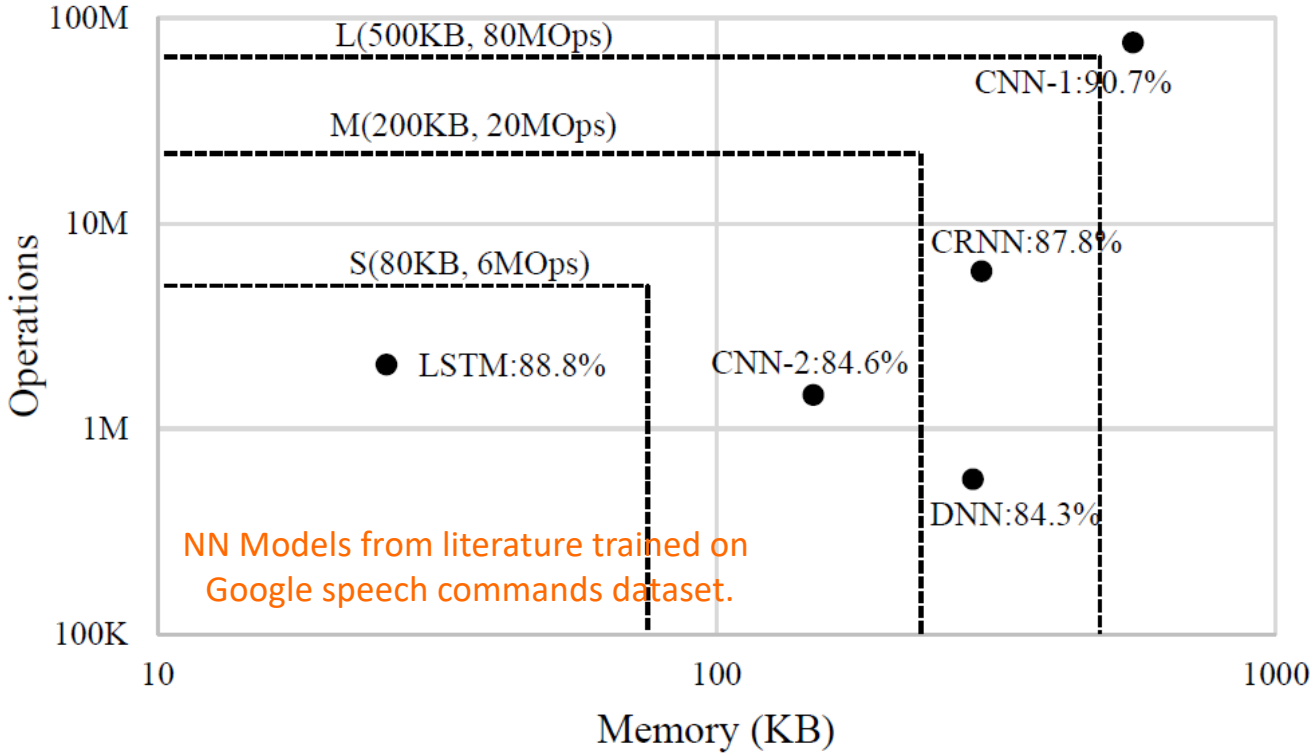
Arm Cortex-M based MCU Platforms



© 2018 Arm Limited

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

Keyword Spotting 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.

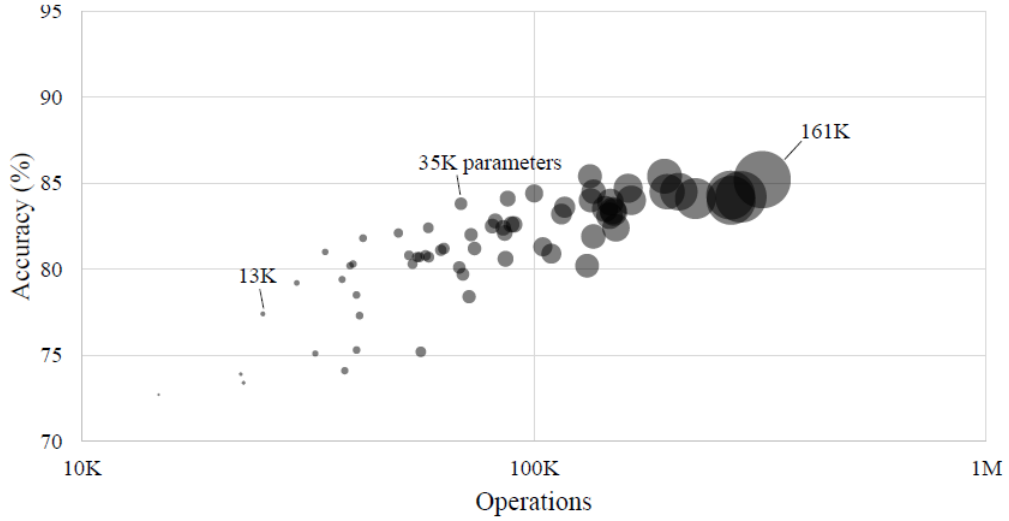
Neural network model search parameters

- 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

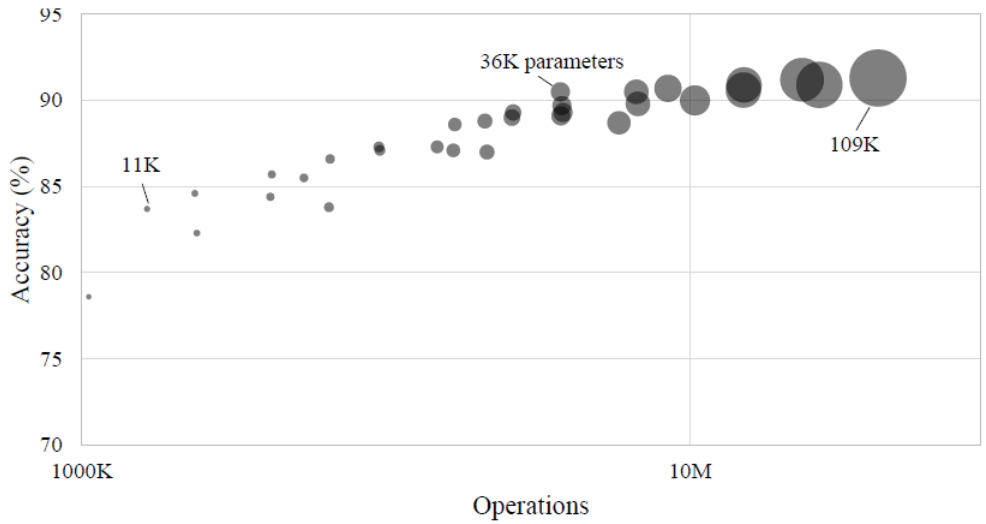
DNN: Google Research (2014): G. Chen et. al. "Small-footprint keyword spotting using deep neural networks"
CNN: Google Research (2015): T. N. Sainath, et. al. "Convolutional neural networks for small-footprint keyword spotting"
LSTM: Amazon (2017): M. Sun, et. al. "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting"
CNN-GRU: Baidu Research (2017): S. O. Arik, et. al. "Convolutional recurrent neural networks for small-footprint keyword spotting"

© 2018 Arm Limited
arm

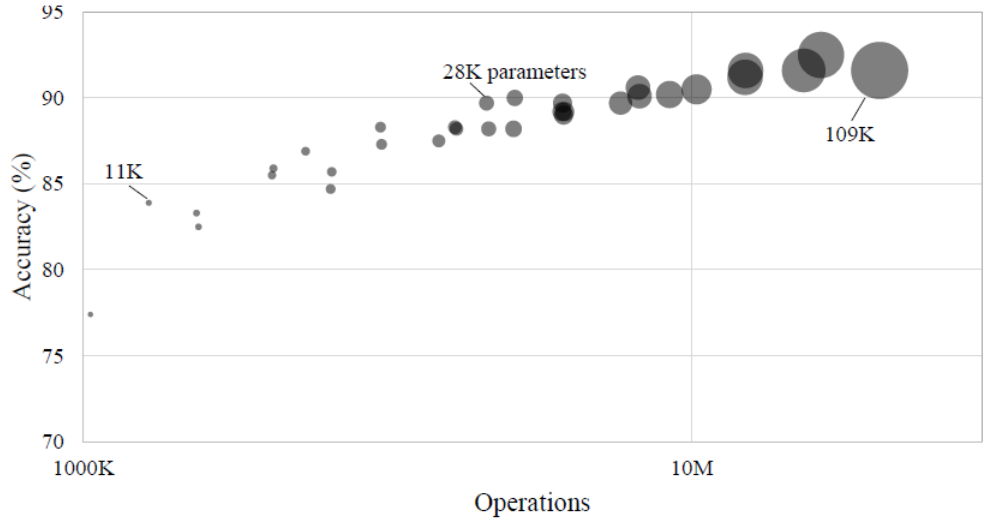
H/W-Constrained NN Model Search



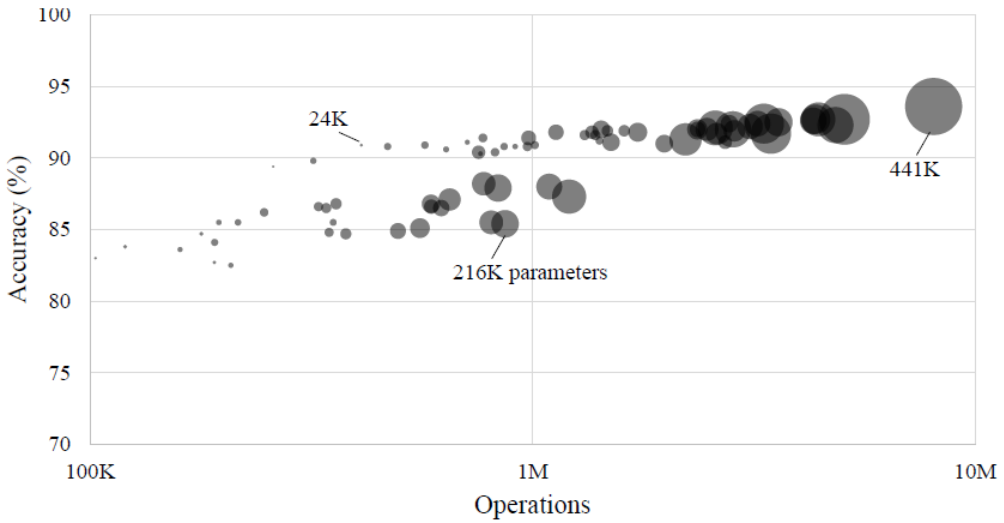
DNN



Basic LSTM



LSTM

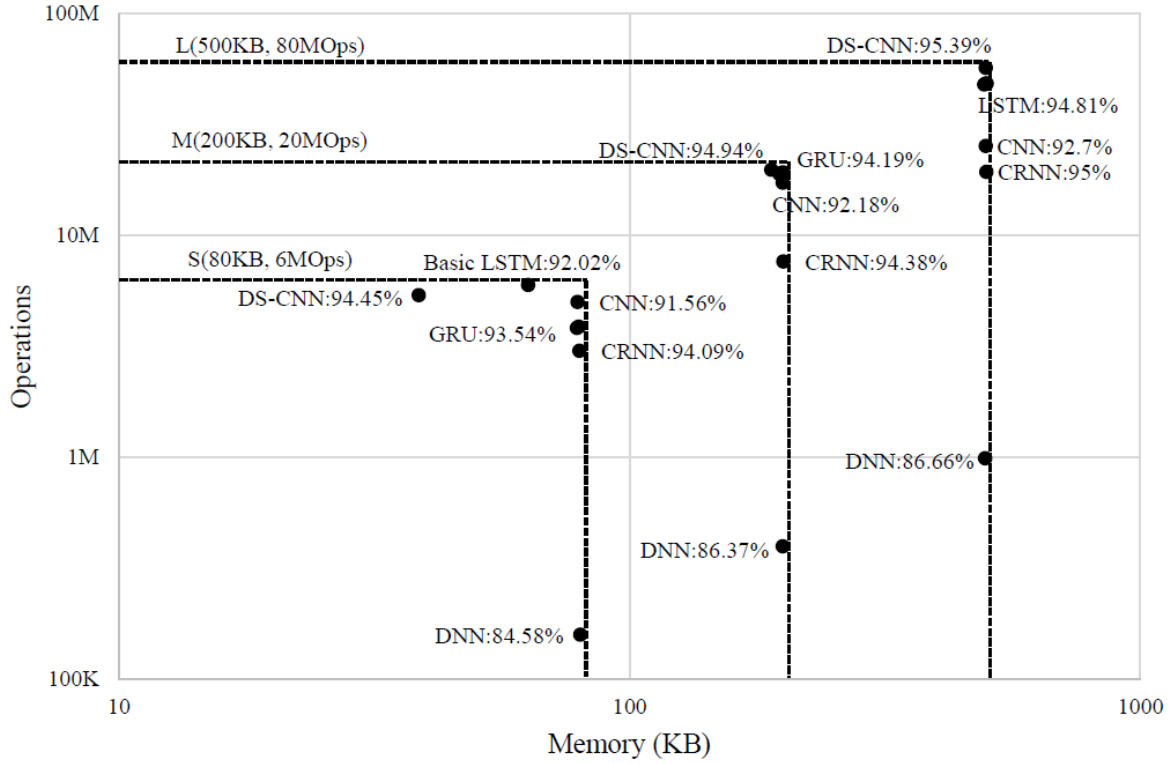


CRNN

arm

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

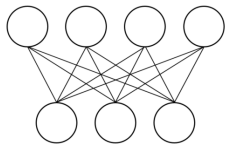
Accuracy asymptotically reaches to 95%.

Y. Zhang, et. al. “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

© 2018 Arm Limited

Developing NN Solution on Cortex-M MCUs



```
// 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

Trained NN model

Deployable Solution

Optimized code on h/w

Hardware-constrained NN model search

NN model translation

Optimized NN functions: CMSIS-NN

© 2018 Arm Limited

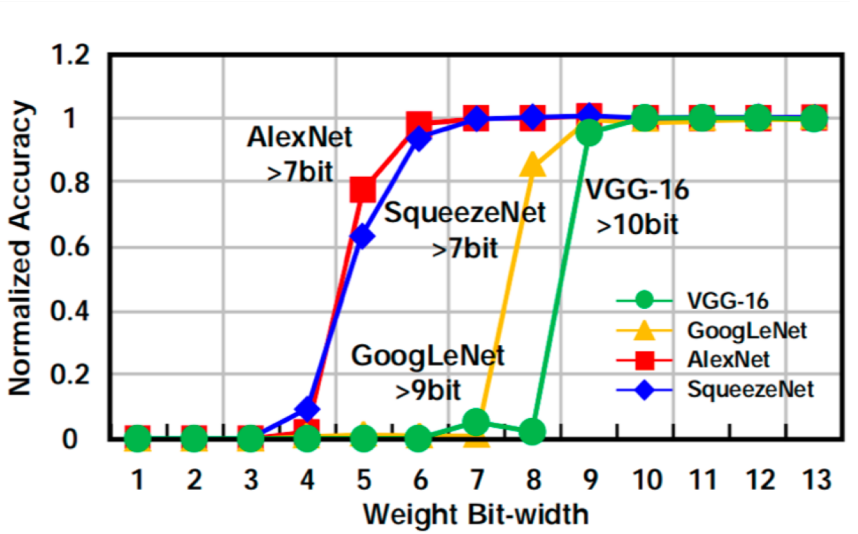
NN Model Quantization

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.

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, because of regularization (or reduce over-fitting).

© 2018 Arm Limited
arm

Model Deployment on Cortex-M MCUs

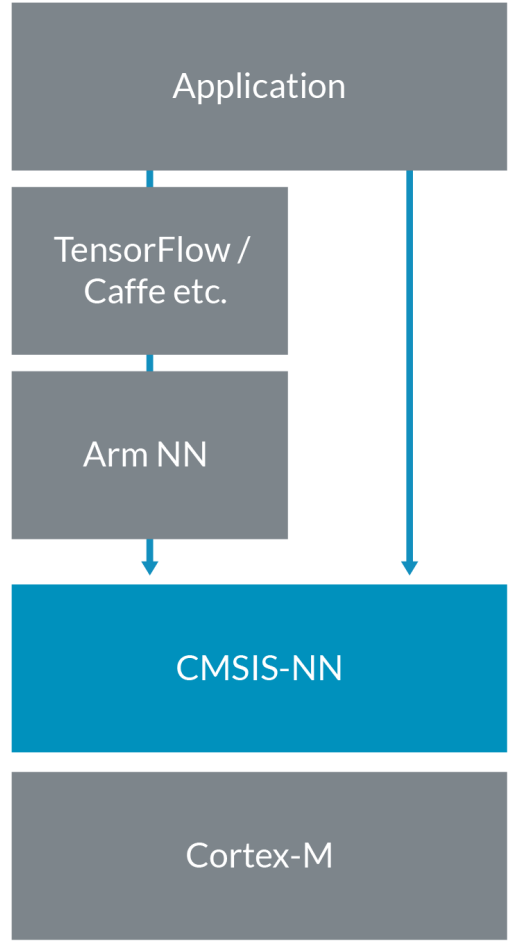
Running ML framework on Cortex-M systems is impractical.

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.



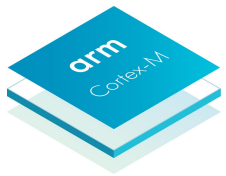
© 2018 Arm Limited
arm

Developing NN Solution on Cortex-M MCUs


```
// 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

Trained NN model

Deployable Solution

Optimized code on h/w

Hardware-constrained NN model search

NN model translation

Optimized NN functions: CMSIS-NN

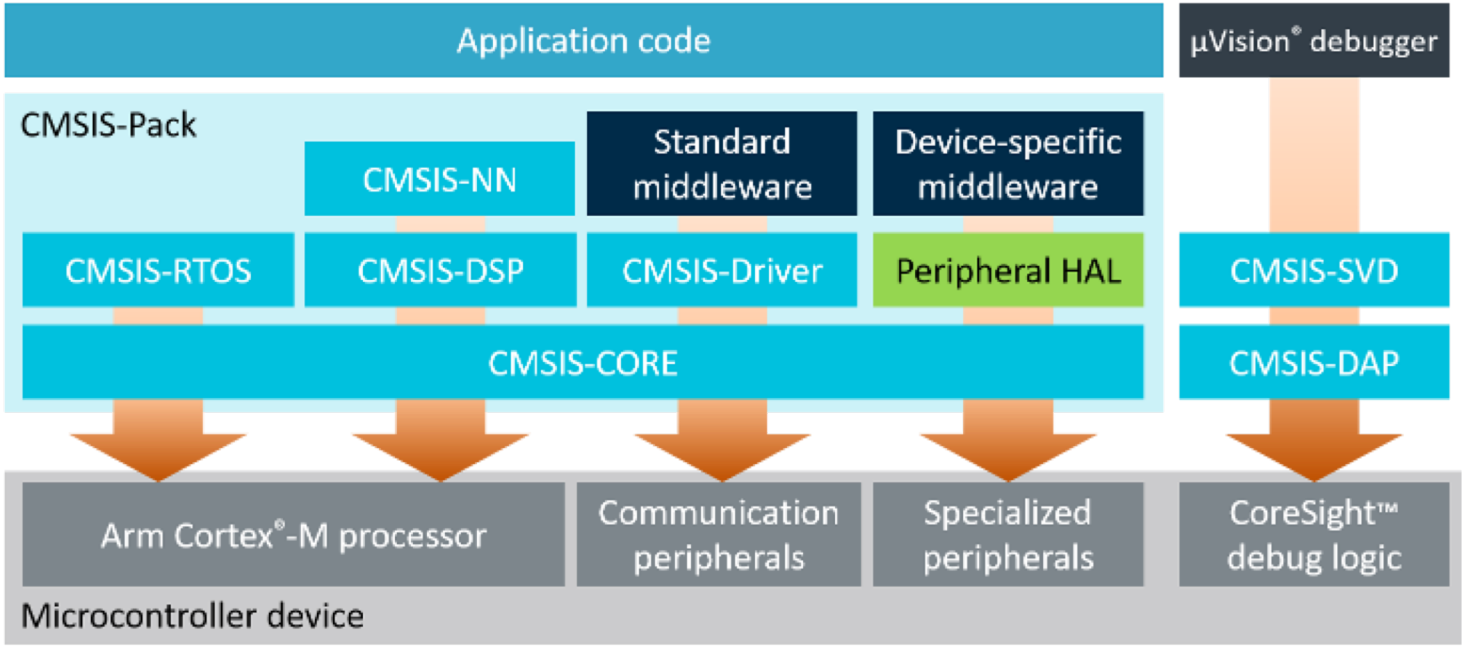
© 2018 Arm Limited
arm

Cortex Microcontroller Software Interface Standard (CMSIS)

CMSIS: vendor-independent hardware abstraction layer for Cortex-M processor cores.

Enable consistent device support, reduce learning curve and time-to-market.

Open-source: https://github.com/ARM-software/CMSIS_5/



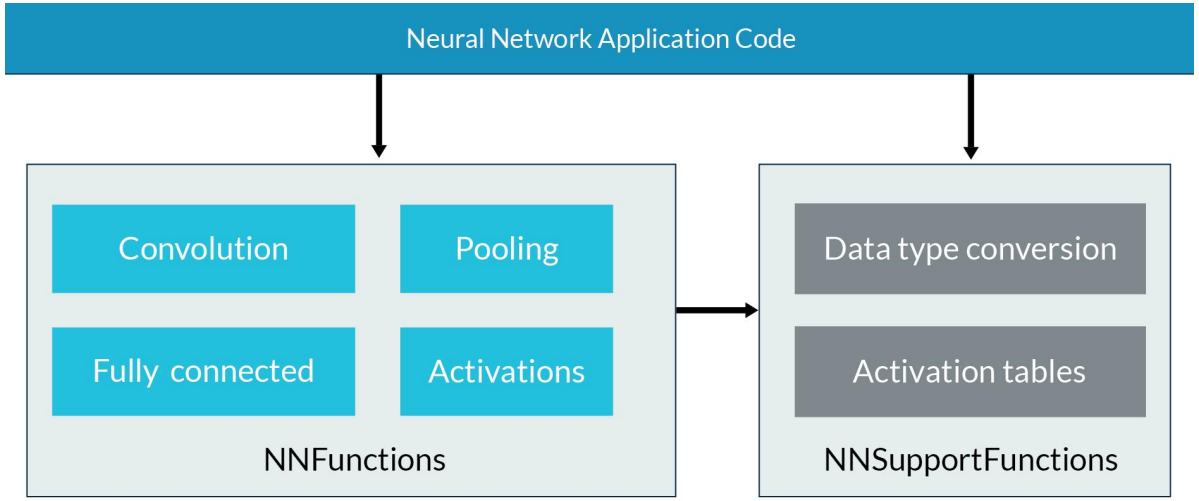
© 2018 Arm Limited
arm

CMSIS-NN

CMSIS-NN: collection of optimized neural network functions for Cortex-M cores

Key considerations

- Improve performance using SIMD instructions.
- Minimize memory footprint.
- NN-specific optimizations: data-layout and offline weight reordering.



© 2018 Arm Limited
arm

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

Convolution

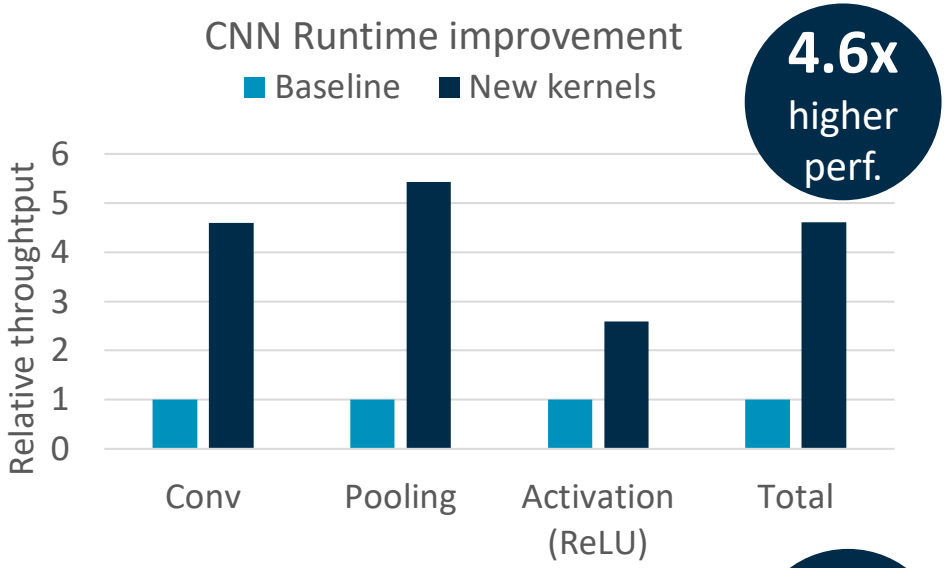
- Boost compute density with GEMM based implementation
- Reduce data movement overhead with depth-first data layout
- Interleave data movement and compute to minimize memory footprint

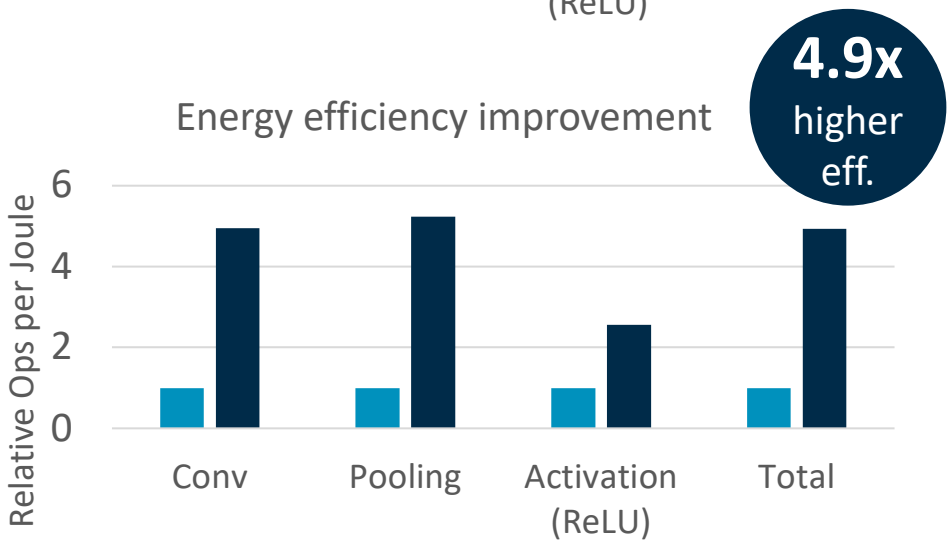
Pooling

- 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





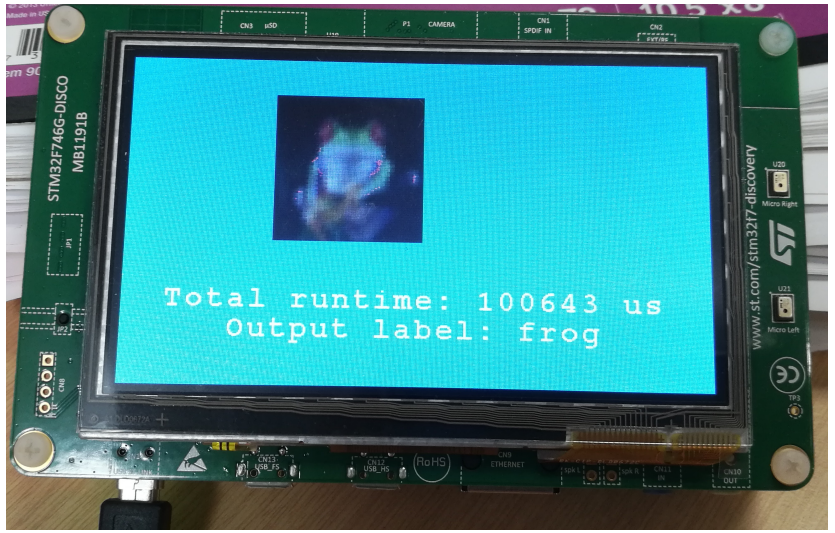
*Baseline uses CMSIS 1D Conv and Caffe-like Pooling/ReLU

© 2018 Arm Limited
arm

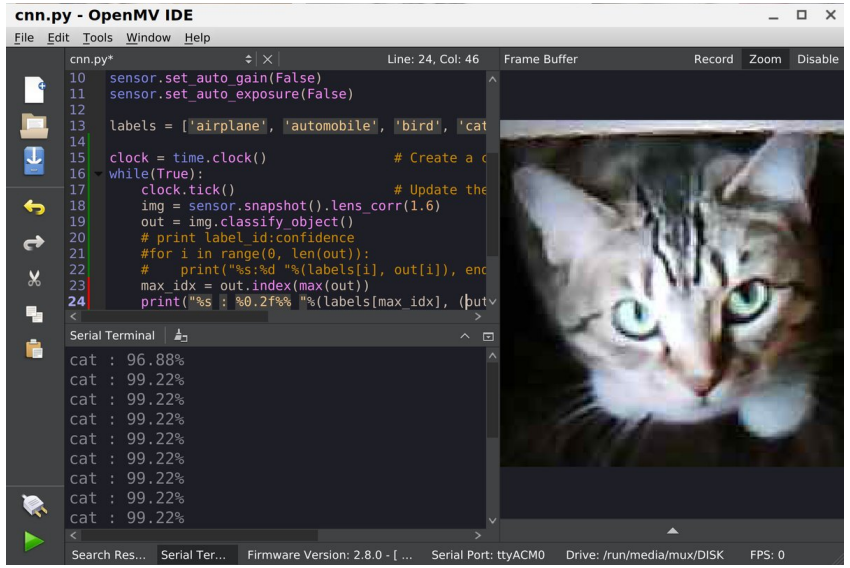
CNN on Cortex-M7

- CNN with 8-bit weights and 8-bit activations
- Total memory footprint: 87 kB weights + 40 kB activations + 10 kB buffers (I/O etc.)
- Total operations: 24.7 MOps, run time: 99.1ms
- Example code available in CMSIS-NN github.

OpenMV platform with a Cortex-M7



NUCLEO-F746ZG: 216 MHz, 320 KB SRAM



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

© 2018 Arm Limited

Demo

https://youtu.be/PdWi_fvY9Og?t=1m

© 2018 Arm Limited

Summary

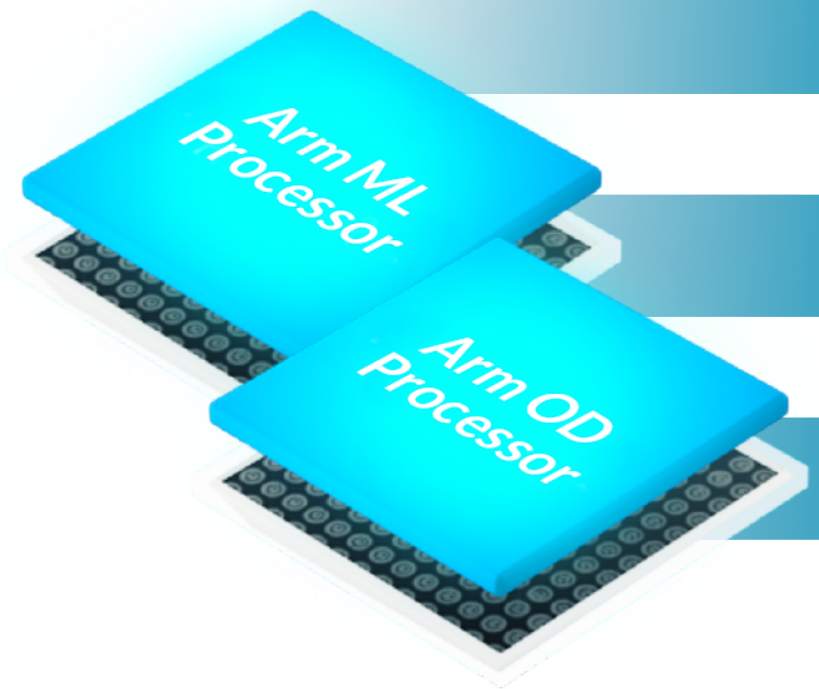
- ML is moving to the IoT Edge and Cortex-M plays a key role in enabling embedded intelligence.
- Case study: keyword spotting on Cortex-M CPU
  - Hardware-constrained neural network architecture exploration
  - Deployment flow of model on Cortex-M CPUs
  - CMSIS-NN: optimized neural network functions

© 2018 Arm Limited
arm

Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!
감사합니다
धन्यवाद

© 2018 Arm Limited

Project Trillium: Arm ML and OD Processors

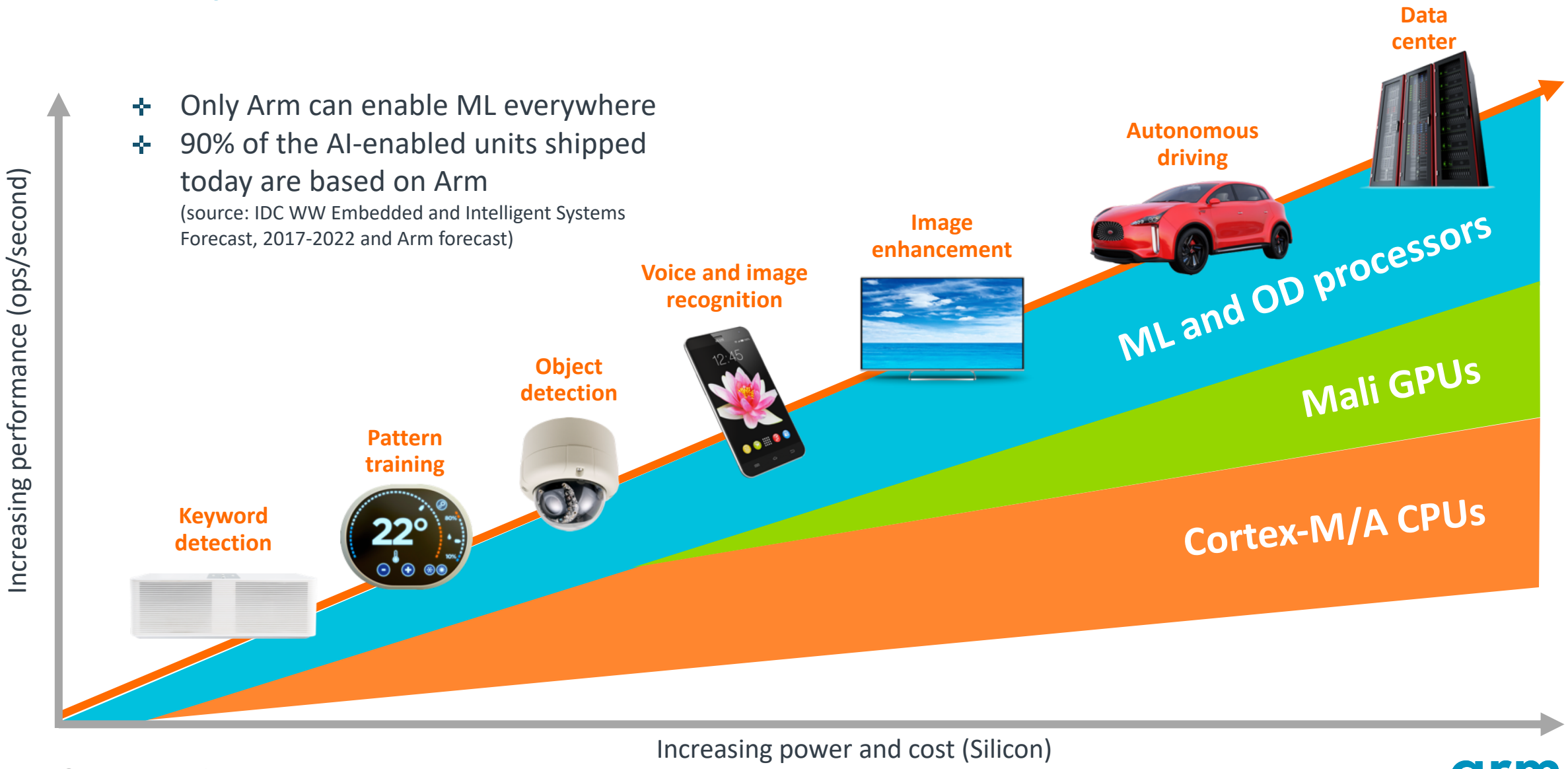


- Ground-up design for high performance and efficiency
- Massive uplift from CPUs, GPUs, DSPs and accelerators
- Enabled by open-source software
- First-generation ML processor targets Mobile market

© 2018 Arm Limited
arm

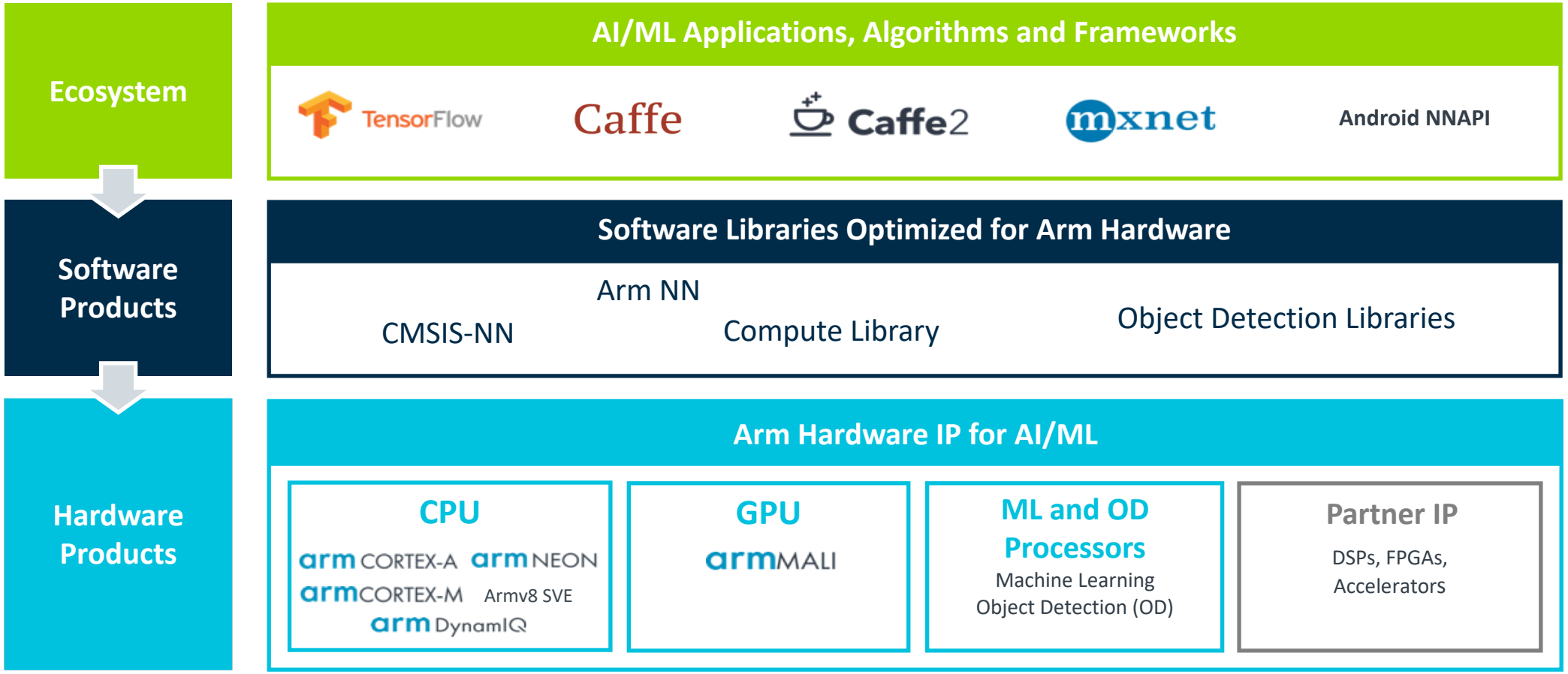
Flexible, Scalable ML Solutions

- Only Arm can enable ML everywhere
- 90% of the AI-enabled units shipped today are based on Arm
  (source: IDC WW Embedded and Intelligent Systems Forecast, 2017-2022 and Arm forecast)



© 2018 Arm Limited
arm

Project Trillium: Arm's ML Computing Platform



© 2018 Arm Limited
arm

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

© 2018 Arm Limited

Thank You!
Danke!
Merci!
谢谢!
ありがとう!
Gracias!
Kiitos!
감사합니다
धन्यवाद

arm

© 2018 Arm Limited

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks

 © 2018 Arm Limited