The Edge Compute Revolution

<Presenter Name & Title>

NXP

May 2019

SECURE CONNECTIONS FOR A SMARTER WORLD

Agenda

- Who is NXP?
- The Machine Learning Revolution
  - Voice, Vision, Vibration
  - Accuracy, inferencing
- NXP solutions for ML/AI
- Further Learning and Q and A

A Position of Strength to Better Serve Our 26,000+ Customers

Employees in
30+ Countries
Headquartered in Eindhoven, Netherlands

~30,000
Employees

9,000
Patent Families

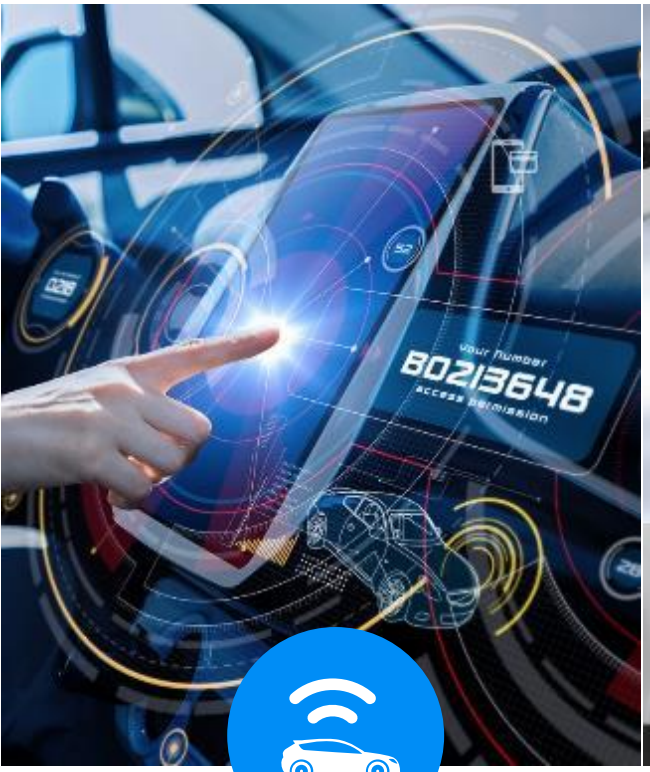
$9.41B
Annual Revenue[1]

60+
Year History

~9,000
R&D Engineers

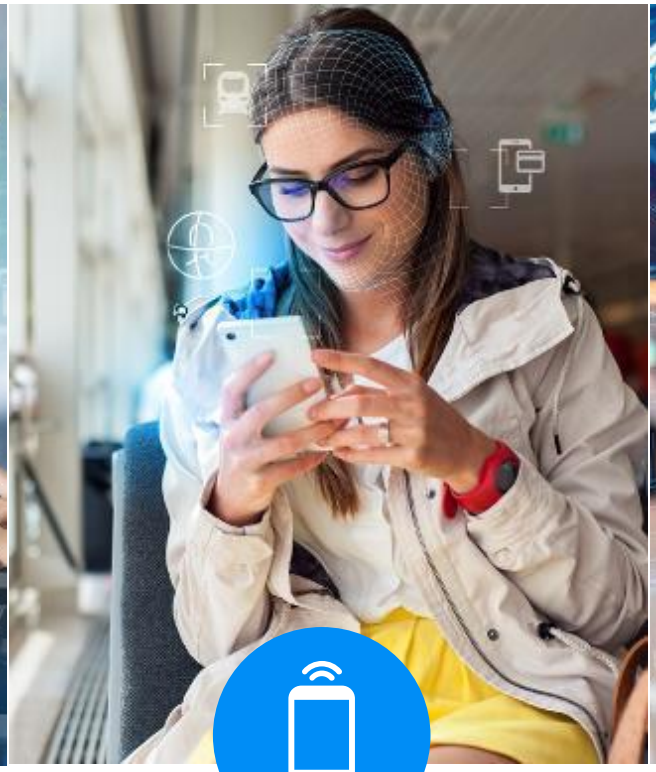
[1] Posted revenue for 2018 – Please refer to the Financial Information page of the Investor Relations section of our website at www.nxp.com/investor for additional information

Our Leadership Focused End Markets



Automotive

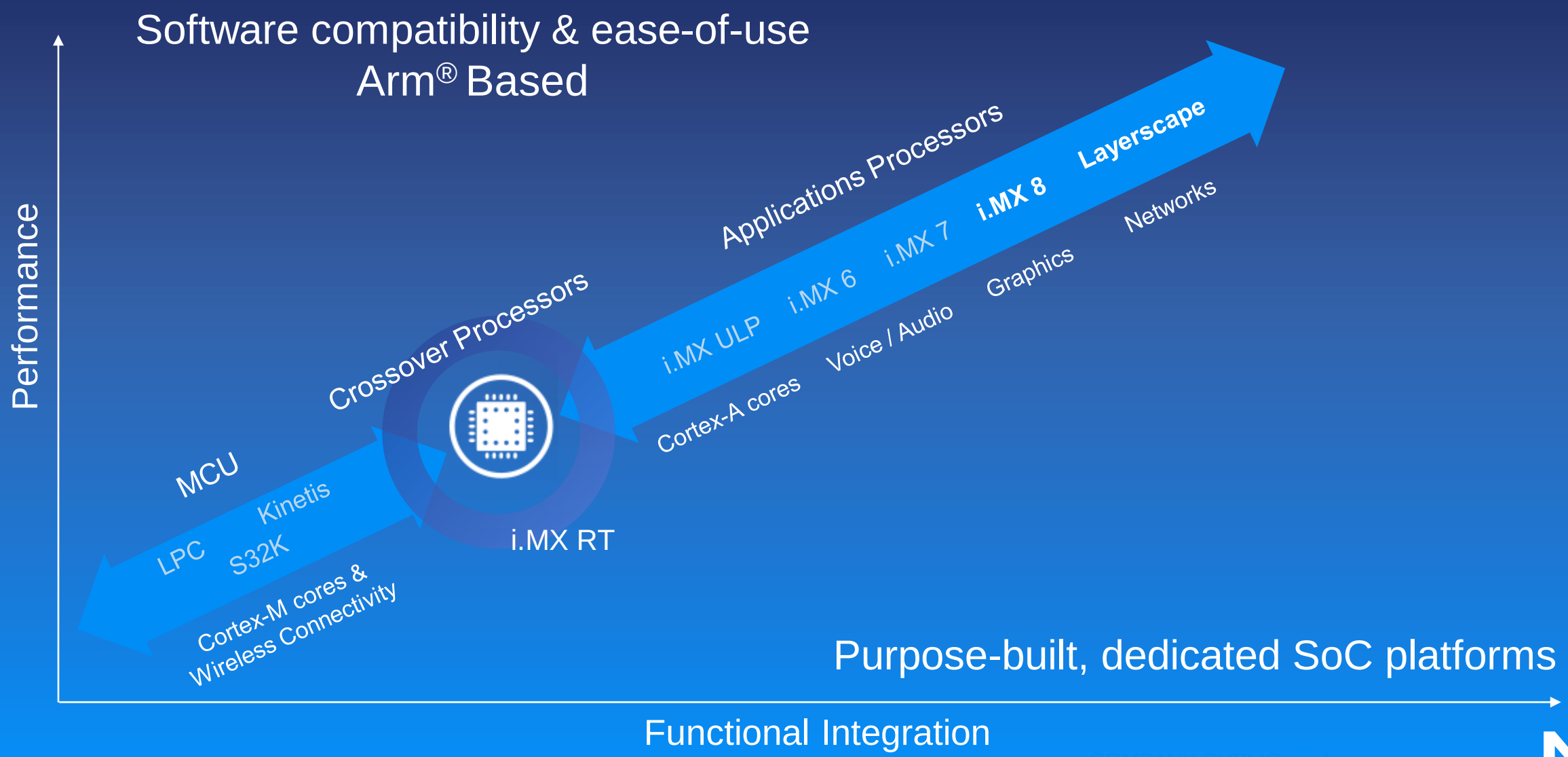
Industrial & IoT



Mobile

Communication Infrastructure

NXP Scalable Processing Continuum



NXP bringing Machine Learning to the Edge

- Support for Machine Learning algorithms via GPU and/or ML HW accelerators and raw compute performance
- All support Trust Architecture for platform security
- Support Linux and Android (i.MX only) distributions
- Optimized for open source ML training and inference models
- Industrial Qualification
- Support long lifecycles

Large-scale video/image processing, data aggregation

i.MX 8QM	LX2160A
• Cortex-A72/ A53 /M4F • Tensilica HiFi4 • 2 x GC7000 (GPUs) • 2x 1G Ethernet, USB, PCI • MIPI-DSI, CSI, HDMI • 8-10W	• Cortex-A72 • 8-16 cores • 2.2GHz • 10/25/40/100 GE • 31W

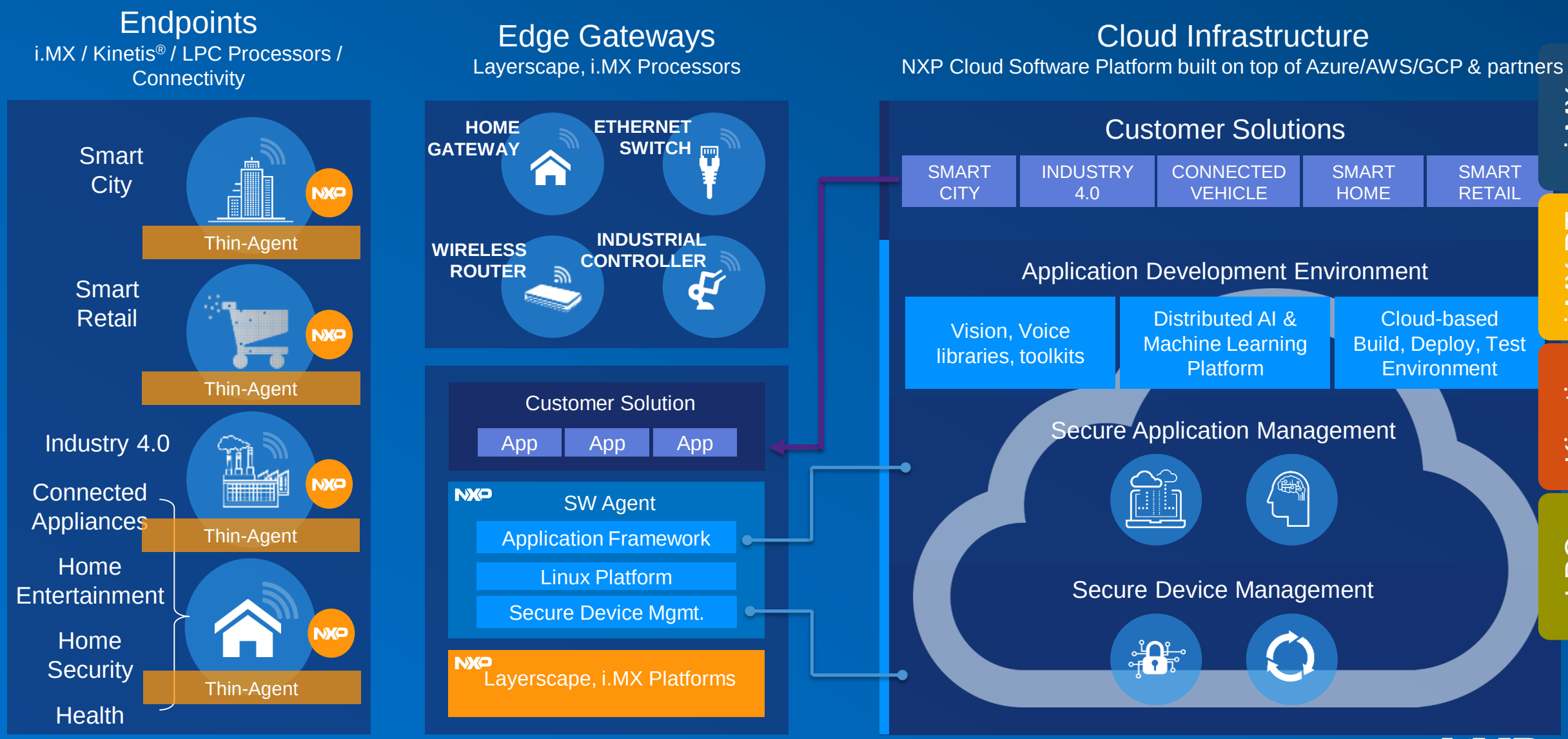
Video/image processing, large-scale analytics, TSN ethernet, Gateway applications

LS1028A	LS1043A	LS1046A	i.MX 8M	i.MX 8M Mini	i.MX 8X
• Cortex-A72 • 2 cores • 1.6GHz • Integrated TSN switch, GPU • 4-9W	• Cortex-A53 • 2-4 cores • 1.6GHz • 1/10G Ethernet, USB, PCI • 5-10W	• Cortex-A72 • 2-4 cores • 1.8GHz • 1/10 G Ethernet, USB, PCI • 10-12W	• Cortex-A53 / M4F • 2-4 A-cores • Up to 1.5 GHz • 3D GPU • MIPI-DSI, CSI, HDMI • Up to 4K HDR (decode) • 2-3W	• Cortex-A53 / M4F • 1-4 A-cores • Up to 1.8 GHz • 2D, 3D GPU • MIPI-DSI, • Video Encode/Decode • 1-2W	• Cortex-A35 / M4F • 2-4 A-cores • Tensilica HiFi4 • 1x GC7000lite • VPU up to 4K h.265 (decode) • 1 G Ethernet • 6-8W

Data acquisition, analytics, monitoring, remote control

i.MX RT1060	LS1012A	LS1021A	i.MX 7D
• Cortex-M7 • 600MHz • 2D graphics, LCD, CSI • Ethernet, USB • 110 uA/MHz	• Cortex-A53 • 1 core • 1GHz • Ethernet, USB, PCI • 1-2W	• Cortex-A7 • 2 cores • 1GHz • Ethernet, USB, PCI • 2W	• Cortex-A7 / M4F • 1-2 A-cores • 1.2GHz • LCD, MIPI-CSI • Ethernet, USB, PCI • .1-.5W

ML requires Edge Computing Infrastructure (Cloud/Gateway/Endpoint)



Machine Learning Revolution

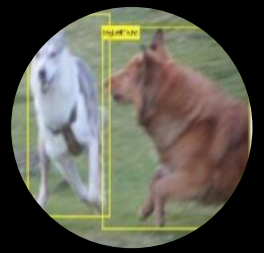
Home Environment

Voice Processing

Gesture Control

Smart Sense & Control

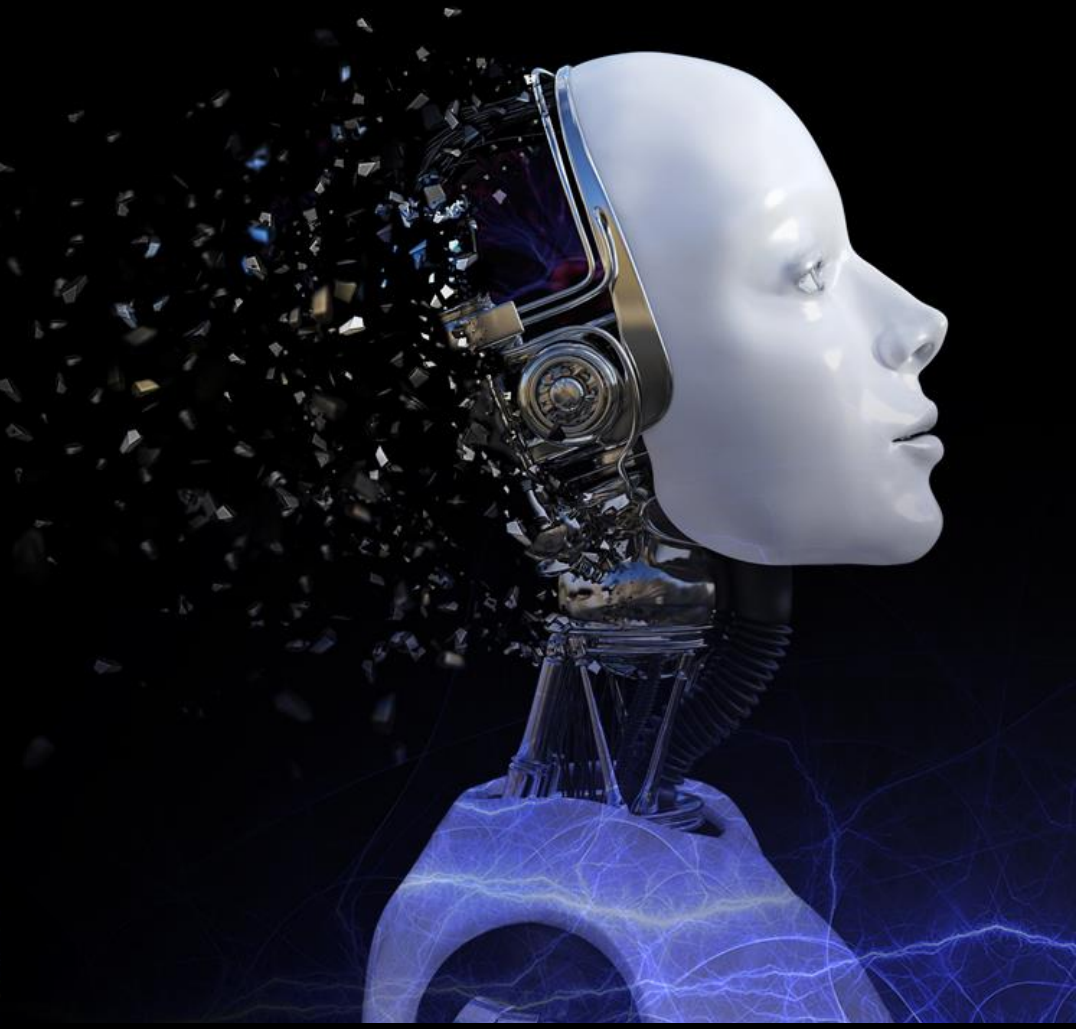
Multi-camera Observation



Personal / Property

Active Object Recognition

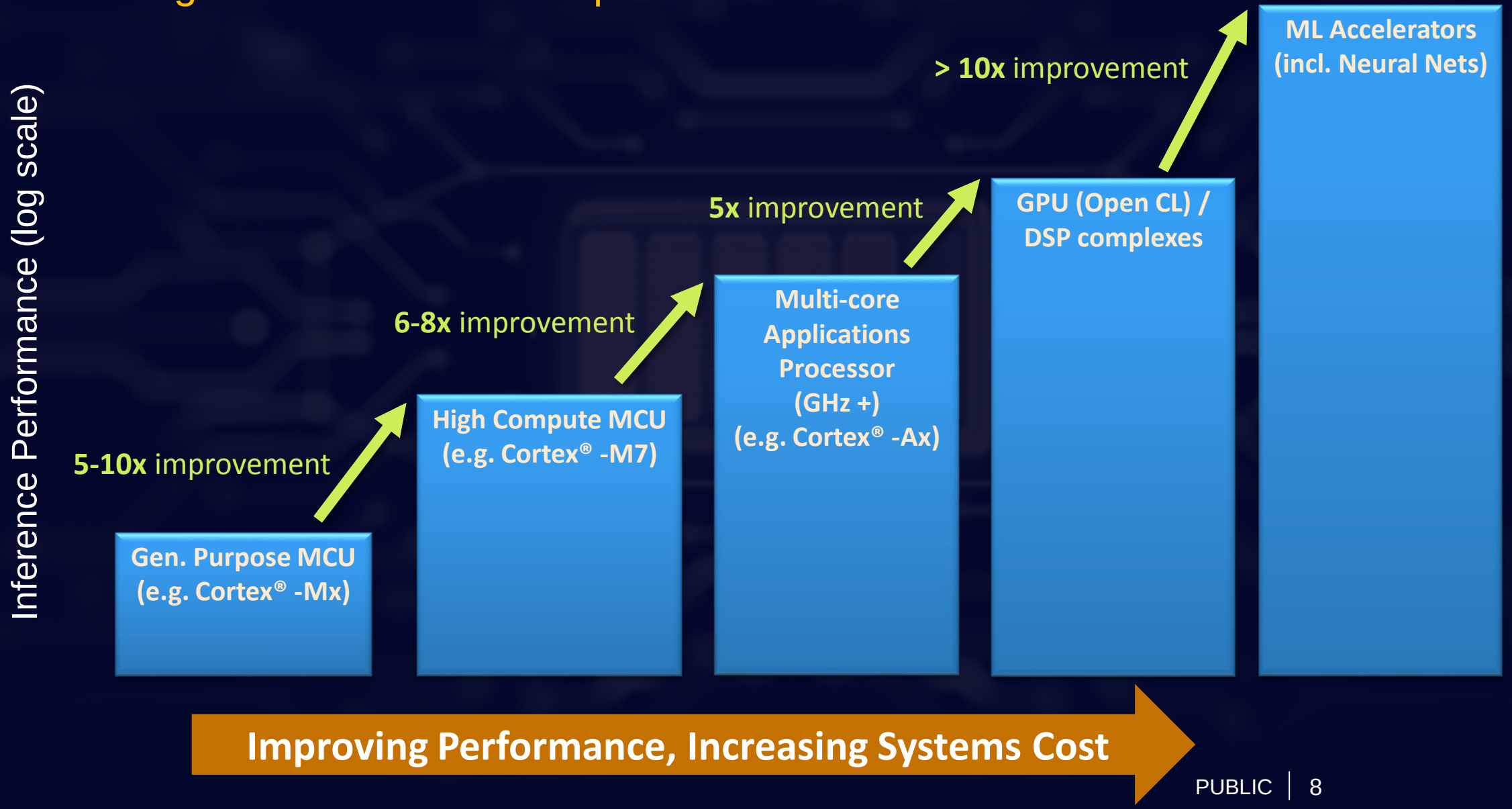
Augmented Reality



Limitless Possibilities

Edge Compute Enabler – Scalable Inference

Balancing cost vs. end-user experience



Artificial Intelligence and Machine Learning

Artificial Intelligence

- The very broad concept of using machines to do “smart” things and act intelligently like a human

Machine Learning

- The concept that if you give machines a lot of data, they can learn how to do smart things on their own.
- Self learning and self improving when give it data

Machine Learning is a subset of Artificial Intelligence

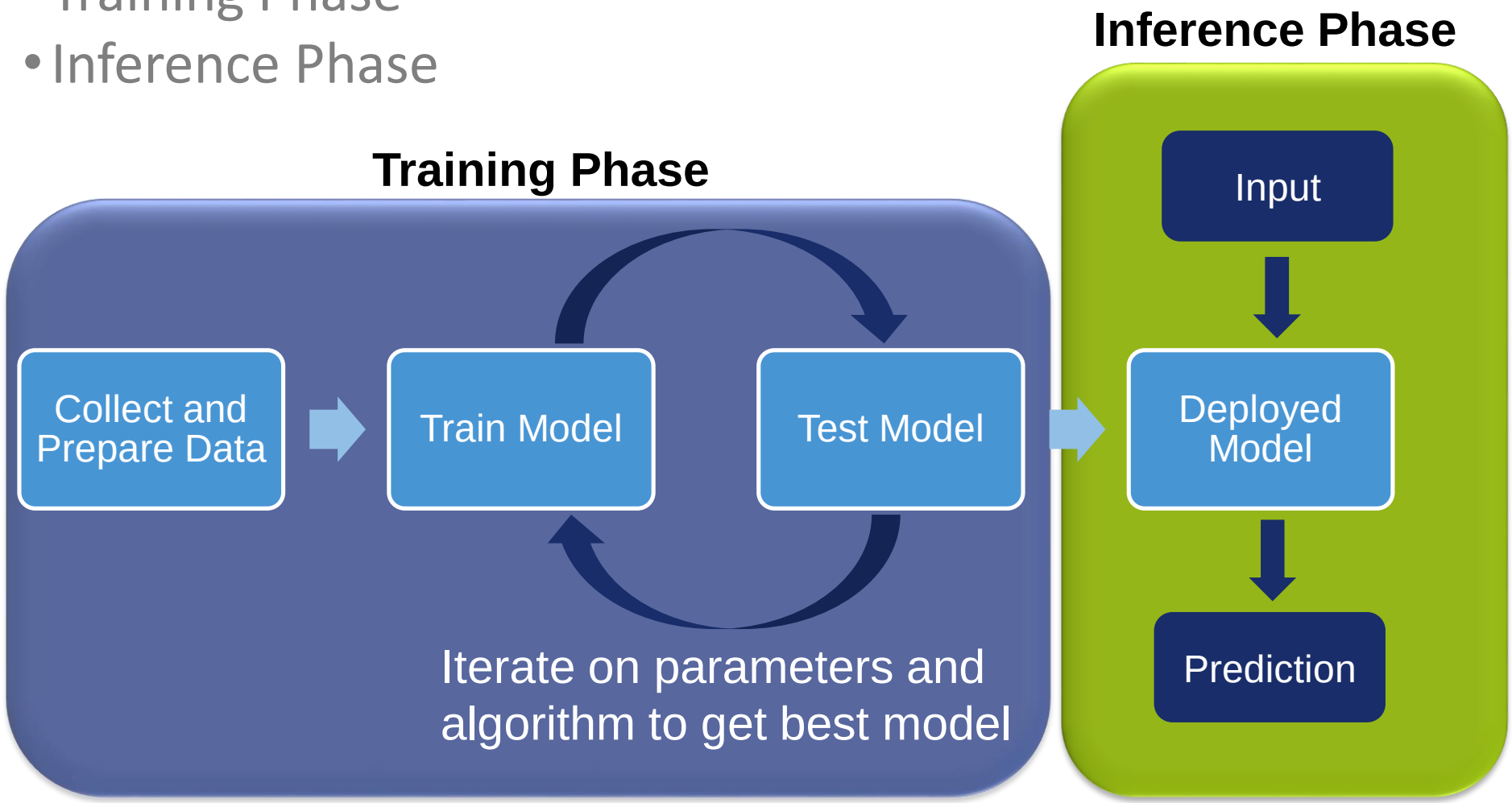
- ML is one of the many possible ways of achieving artificial intelligence
- One of the most popular and promising areas of AI today

Deep Learning is a subset of Machine Learning

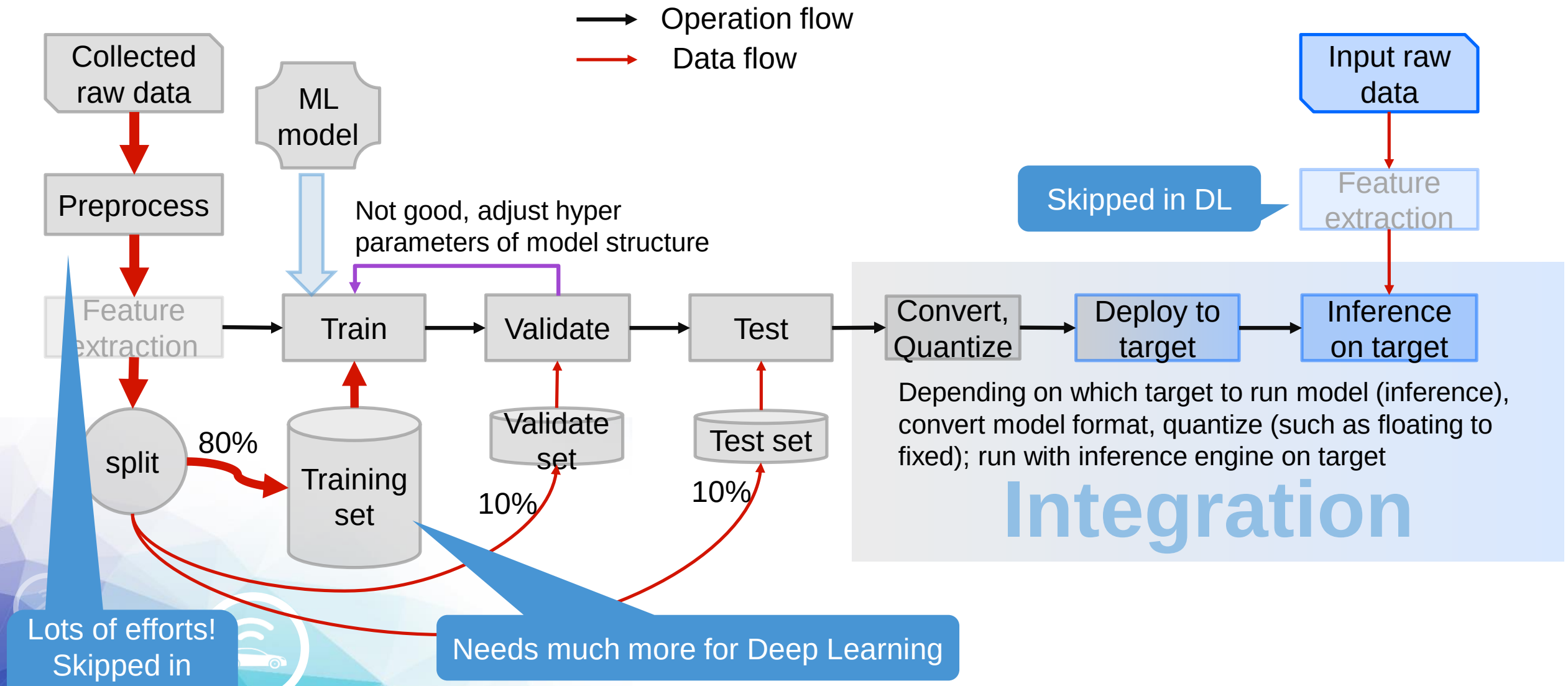
- Uses Neural Networks to do the learning that ML requires
- Automatically determines most relevant data aspects to analyze instead of having to be explicitly told
- Needs lots and lots of data
- There are other methods of doing Machine Learning, but the Deep Learning method shows the most promise

Machine Learning Process

- Training Phase
- Inference Phase



From data collection to inferencing



Simplified Neural Network Model

Input Layer

Hidden Layer(s)

Output Layer

0.01 Armadillo

0.05 Dog

0.93 Cat

0.01 Dodo

Node Value $a_x = ReLU(w_1a_1 + w_2a_2 + w_3a_3 \ldots . - b)$

Weights (w_x)

Bias

Model

Model Frameworks and Formats

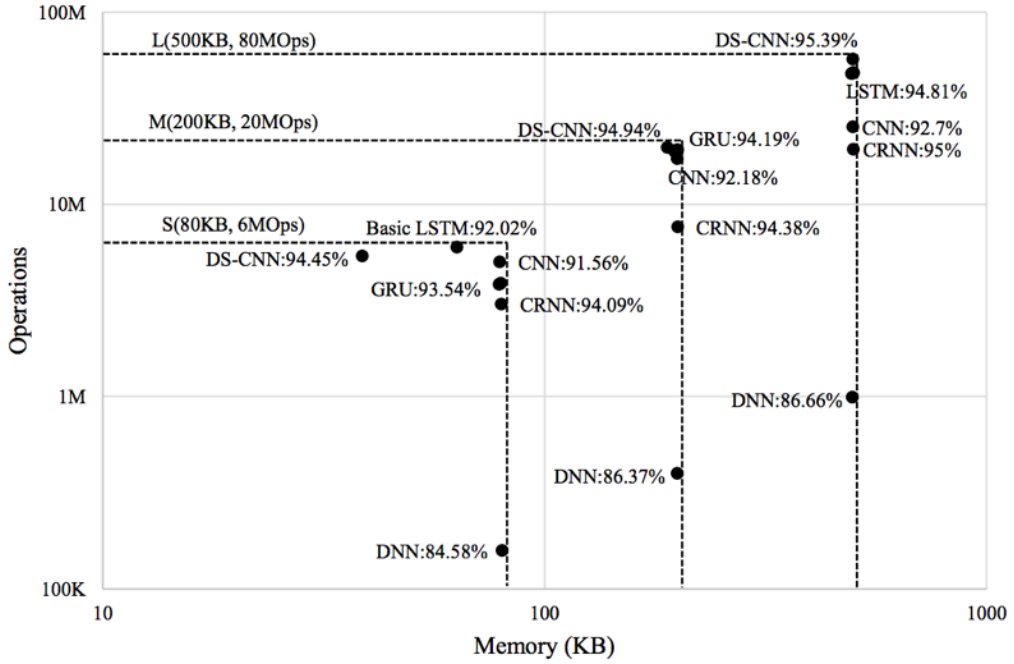
- There are several popular model frameworks in use today.
- Each framework has their own APIs and methodologies for building and training a model
- This is a constantly changing list as new software is released:
  - **TensorFlow** – Google framework
  - **Keras** – higher level API, usually built on top of TensorFlow
  - **Caffe2** – Facebook framework
  - **PyTorch** – Facebook framework
  - **ONNX**- open neural net exchange format

- Python is basis for most ML frameworks
- New breakthroughs constantly and favorite framework du jour can change quickly.

Machine Learning Accuracy

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



- Depth wise 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: github.com/ARM-software/ML-KWS-for-MCU

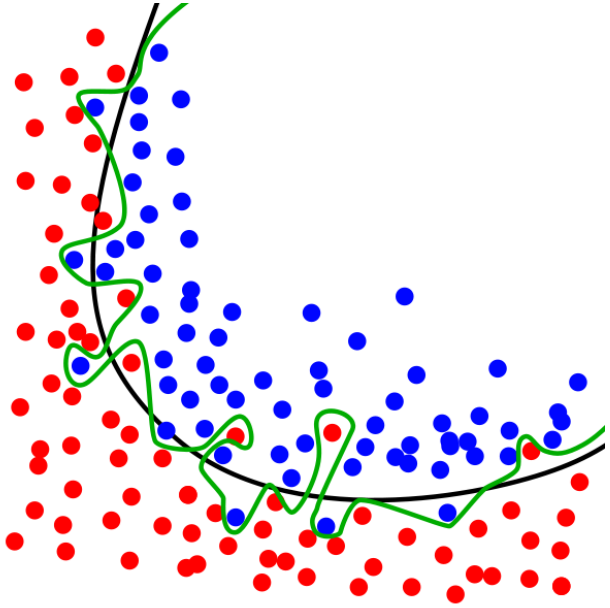
Model Accuracy Continued

- Models are not perfect. Especially when scaling down models to fit on embedded systems.

Model	Million MACs	Million Parameters	Top-1 Accuracy	Top-5 Accuracy
MobileNet_v1_1.0_224	569	4.24	70.9	89.9
MobileNet_v1_1.0_192	418	4.24	70.0	89.2
MobileNet_v1_1.0_160	291	4.24	68.0	87.7
MobileNet_v1_1.0_128	186	4.24	65.2	85.8
MobileNet_v1_0.75_224	317	2.59	68.4	88.2
MobileNet_v1_0.75_192	233	2.59	67.2	87.3
MobileNet_v1_0.75_160	162	2.59	65.3	86.0
MobileNet_v1_0.75_128	104	2.59	62.1	83.9
MobileNet_v1_0.50_224	150	1.34	63.3	84.9
MobileNet_v1_0.50_192	110	1.34	61.7	83.6
MobileNet_v1_0.50_160	77	1.34	59.1	81.9
MobileNet_v1_0.50_128	49	1.34	56.3	79.4
MobileNet_v1_0.25_224	41	0.47	49.8	74.2
MobileNet_v1_0.25_192	34	0.47	47.7	72.3
MobileNet_v1_0.25_160	21	0.47	45.5	70.3
MobileNet_v1_0.25_128	14	0.47	41.5	66.3

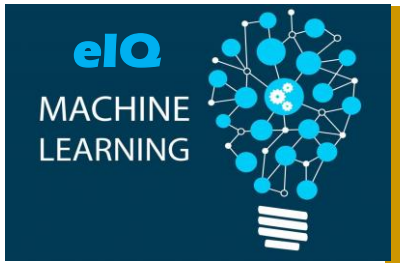
Overfitting

- Want to ensure model does not overfit on training data so that it then fails on unseen data
- Model needs to generalize the training data
  - This means model will not always give 100% confidence, even on the data a model was trained on



NXP Solutions for the Smarter Edge

NXP broad-based machine learning solutions and support (available TODAY!)



eIQ™ Machine Learning Enablement

- eIQ (edge intelligence) is a collection of libraries and dev tools for building ML applications targeting NXP microcontrollers and apps processors
- Use case: general purpose optimized edge AI/ML inference enablement
- i.MX 8 family (GA w/ 4.19 release), i.MX RT1050/1060 (GA w/ 2.6 release)
    - Includes sample end-to-end applications

Third Party SW and HW

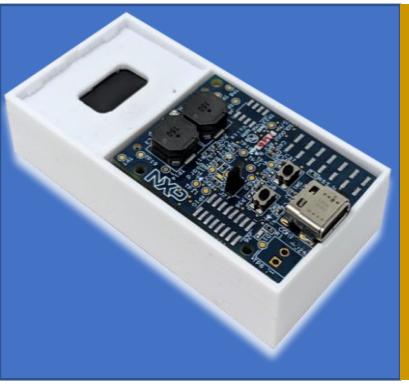
Coral Dev Board
- Owned by Google's Coral team; built on i.MX 8M and Google TPU devices
- Supported by NXP and Google Coral team
- Use case: development kit - link

i.MX 8M Development Kit for Amazon® Alexa Voice Service w/ DSP Concepts
- Supported by the i.MX Community
- Use case: development kit - link

Au-Zone Development Tools
- Use case: general-purpose AI/ML enablement

EdgeScale™ Solution

Secure deployment of applications (incl. AI/ML) through docker containers
Cloud dashboard to securely enroll Edge devices, monitor their health, attest and deploy container applications and firmware updates.
Currently Layerscape® devices; plans to add i.MX this year



Turnkey Solutions

AVS Solution (Alexa Voice Services)
- i.MX RT106A (part# SLN-ALEXA-IOT) Link
- Enabled with OTA, execution from encrypted flash, far-field, Amazon® wake-word, Alexa for MCU client (NXP developed)
- Use case: voice-based IoT products for consumer applications

Other turnkey solutions targeting vision and vibration (anomaly detection), coming soon for broad market
Anomaly detection and facial recognition solutions based on i.MX RT, i.MX 8M Mini
Contact Distribution or NXP sales for early access.

Open “Source” – Lots of Options

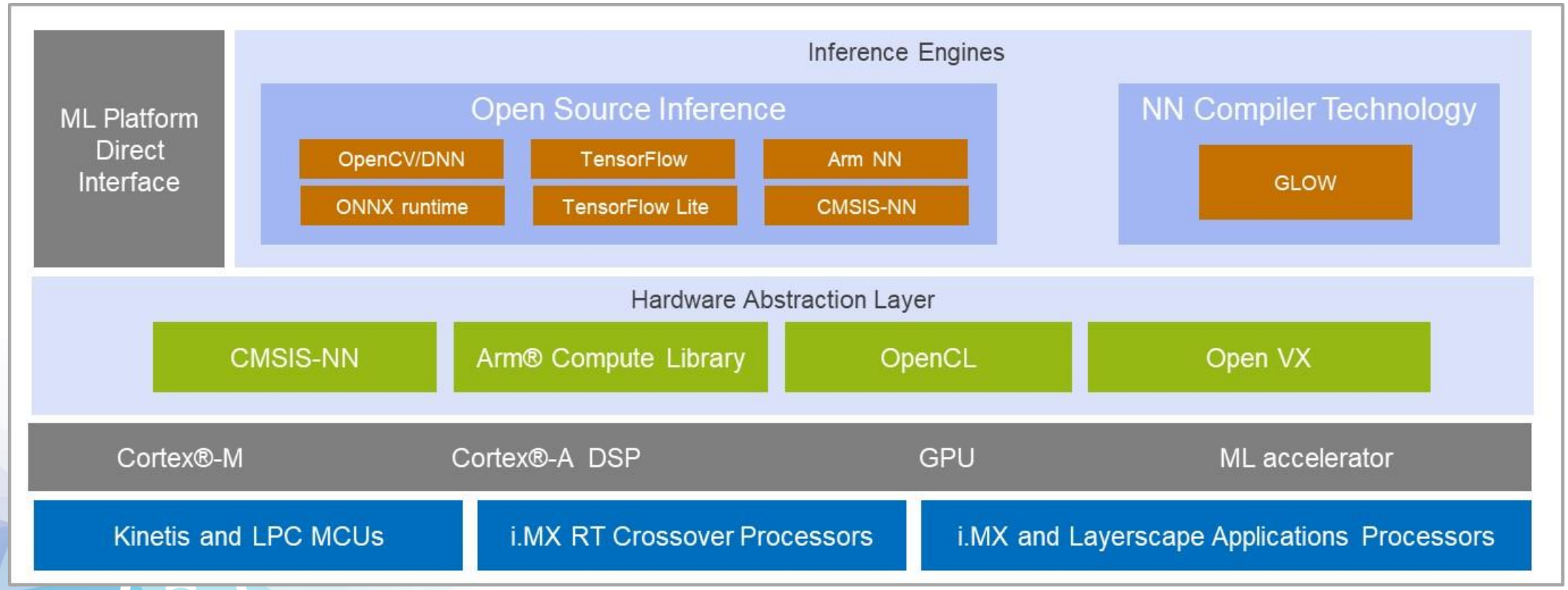
What is eIQ?

eIQ is a software platform for NXP microprocessors and microcontrollers
to do inference of models on embedded systems

- Tools currently under development include:
  - Inference engines: **Arm NN, OpenCV, Arm CMSIS-NN, TensorFlow Lite,** and more
    - May require converting a pre-trained model into a format that eIQ supports
  - **End-to-end example applications** demonstrating typical customer use-cases (e.g. camera → inference engine)
  - Support for **emerging neural net compilers** like GLOW
  - Suite of classical ML algorithms such as support vector machine (SVM) and random forest
  - Supported tools and application examples will differ between i.MX apps and i.MXRT
- Integrated as middleware into standard **NXP Yocto and MCUXpresso SDK releases**
- End goal is to enable customers to develop embedded applications that can utilize machine learning models

eIQ™ – all the ingredients to deploy an inference engine

eIQ is a software platform for NXP microprocessors and microcontrollers to do inference of models on embedded systems



eIQ Components Phased Development Approach

We are here

Enable

1. Feasibility
   - Open Source Components
   - Benchmarking
   - Potential to Optimize/Advance
2. Port
   - Apps processors and/or MCUs
   - Build recipes/makefiles
   - Test examples
3. Deliver (MCUXpresso, Yocto)

Receive Customer feedback, Improve

Optimize

1. Optimize Open Source Components
   - Support community process
   - NXP proprietary mechanisms
2. Model optimization (off-line)
   - Quantization, pruning, compression
3. eIQ Sample Applications
4. Deliver (MCUXpresso, Yocto)

Receive Customer feedback, Improve

Advanced

1. Optimize ML Functions/ Kernels
   - GPU
   - DSP
   - MLa
2. Heterogenous Compute
3. Advanced model optimization
4. Deliver (MCUXpresso, Yocto)

Receive Customer feedback, Improve

eIQ for i.MX

eIQ for i.MX Apps Processors – EAR1

- **Two types of inference engines currently available for i.MX in EAR1:**
  - **OpenCV 3.4.1:** Image processing, video encoding/decoding, video analysis, object detection, and processing of neural networks. Utilizes Arm NEON for acceleration. GPU support coming.
  - **Arm NN 18.08:** Open source tool from Arm. Current version utilizes Arm Compute Library to optimize neural network operations running on Cortex A cores. Utilizes Arm NEON for acceleration.
  - Coming Soon: **ONNX run-time, TensorFlow Lite**

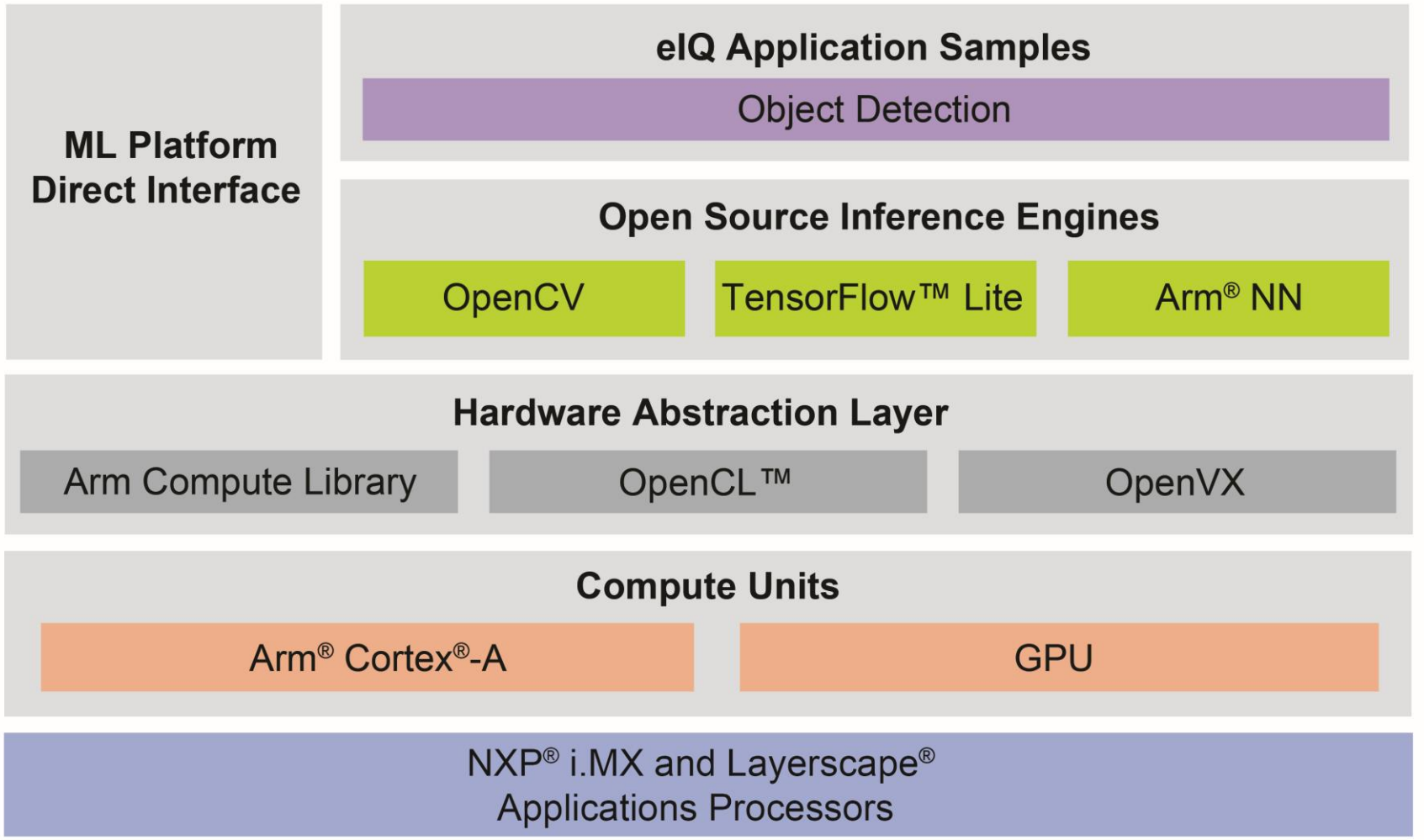
eIQ	Scope	NN ML	Non-NN ML	Supported Accelerators	
				Arm Neon	GPU
Arm NN	NN Machine Learning Library	✓	×	✓	Not supported
OpenCV	Computer Vision Library	✓	✓	✓	Not tested

eIQ	Supported NN Formats			
	Caffe	**TensorFlow**	**TensorFlow Lite**	**ONNX**
Arm NN	✓	✓	✓	✓
OpenCV	✓	✓	×	×

Supported Platforms

- ✓ i.MX 8QM
- ✓ i.MX 8X
- ✓ i.MX 8M
- ✓ i.MX 8M Mini
- ✓ i.MX 7ULP (coming soon)
- ✓ Layerscape (coming soon)

eIQ MPU Implementation



eIQ for i.MX Apps Processors – EAR1

The following supported models can be used by customers as a starting point for developing ML applications:

Arm NN 18.08 SDK

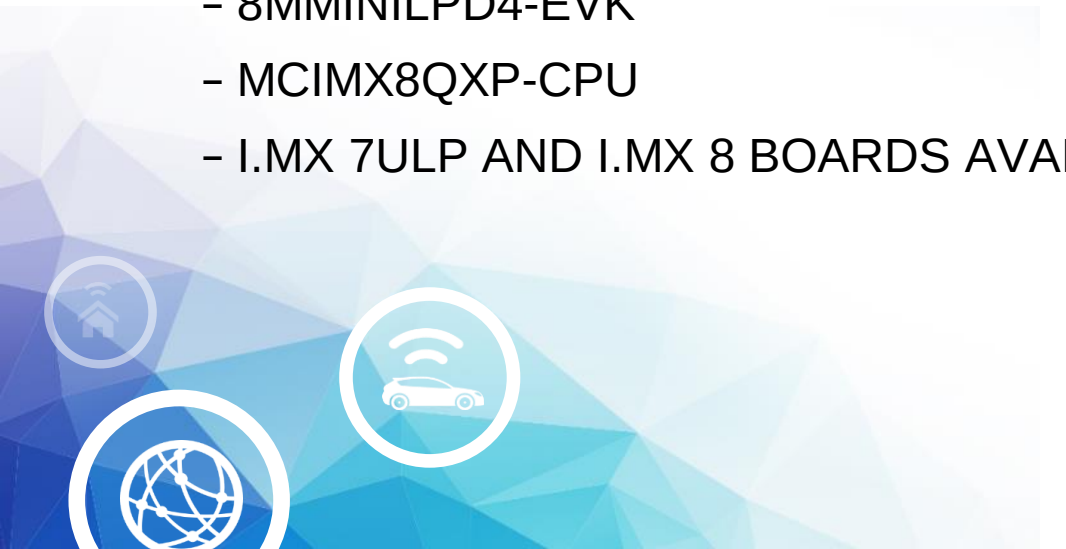
- **Caffe models**
  1. bvlc_alexnet.caffemodel
  2. Inception21k.caffemodel
  3. lenet_iter_9000.caffemodel
  4. ResNet_50_ilsvrc15_model.caffemodel
  5. VGG_CNN_S.caffemodel
- **TensorFlow models**
  1. inception_v3_inf_graph.pb
  2. simple_mnist_tf.prototxt
  3. mobilenet_v1_1.0_224_eval.pbtxt
- **TensorFlow Lite model**
  1. mobilenet_v1_1.0_224_quant.tflite
- **ONNX models**
  1. mnist_onnx.onnx
  2. mobilenetv2-1.0.onnx

OpenCV 3.4.1 SDK

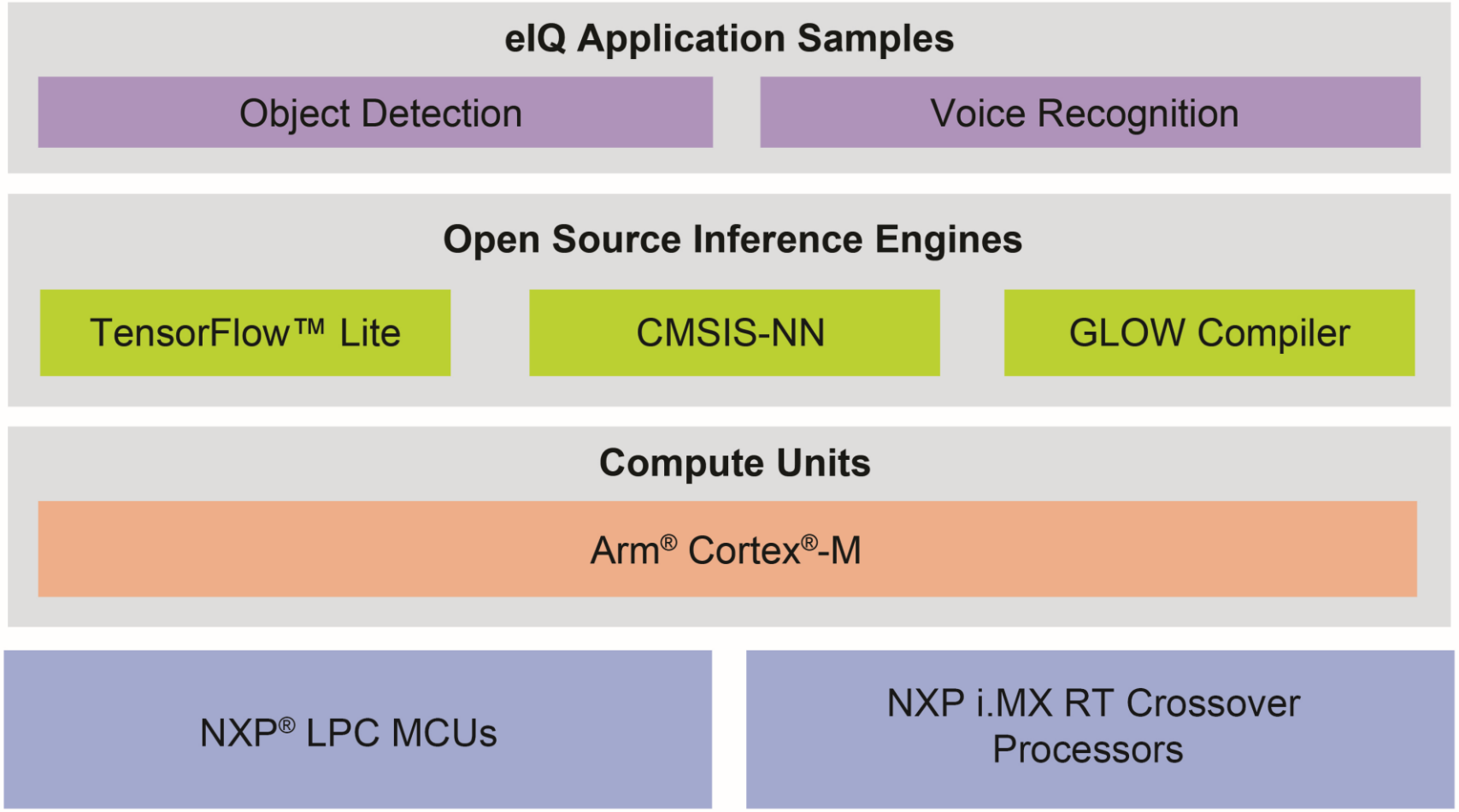
- **DNN models**
  1. caffe_googlenet – GoogLeNet in Caffe
  2. tf_inception – Inception(GoogLeNet) in TensorFlow
  3. ssd_mobilenet_object_detection – Object detection with MobileNet based SSD
  4. resnet_ssd_face – Face detection with ResNet based SSD
  5. ssd_object_detection – Object detection with VGG based SSD
  6. yolo_object_detection – Object detection with YOLO
  7. faster_rcnn – Object detection with Faster R-CNN
  8. colorization – Image colorization
  9. fcn_sgmem – Image segmentation with FCN
- **Non-NN algorithms**
  1. SVM
  2. Principal Component Analysis
  3. Logistic regression

Get Started: eIQ on i.MX MPU

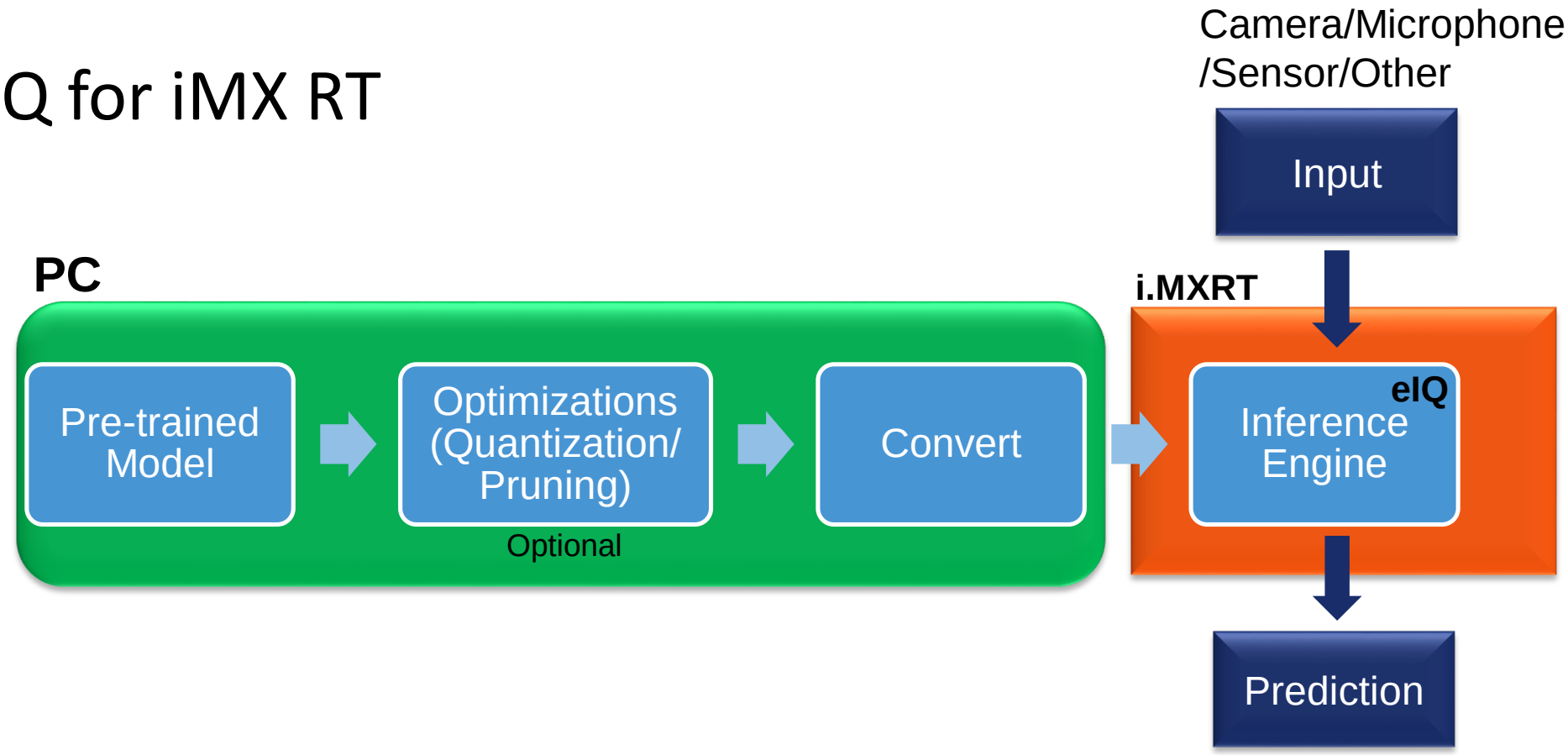
- Download eIQ EAR1 (Now)
  - https://community.nxp.com/docs/DOC-343013
- Download eIQ (Later)
  - Full integration into Yocto Linux SDK in September
- Order your board from Future Electronics
  - MCIMX8M-EVK
  - 8MMINILPD4-EVK
  - MCIMX8QXP-CPU
  - I.MX 7ULP AND I.MX 8 BOARDS AVAILABLE IN JUNE



eIQ MCU Implementation



eIQ for iMX RT



- For i.MX RT there are several different types of inference engines available in eIQ:

  TensorFlow Lite – Used for TensorFlow model frameworks

  CMSIS-NN – Can be used for several different model frameworks

  Glow – Machine Learning compiler (Coming Soon)

eIQ Examples

Three ML application examples available:

	CIFAR-10	Keyword Spotting (KWS)	Label Image
Description	Classifies 32x32 image into one of 10 categories using the CIFAR-10 dataset	Detects specific keywords from pre-recorded audio sample	Classifies an image into one of 1000 categories
TensorFlow Lite Example			
CMSIS-NN Example			

- TensorFlow Lite examples currently support MCUXpresso IDE and IAR
- CMSIS-NN examples currently support MCUXpresso IDE, IAR, and Keil

Get Started: eIQ on i.MX RT

- Download eIQ EAR1 (Now)
  - eIQ iMX Getting Started
  - Download eIQ (Later)
    - Full integration into MCUXpresso SDK in June
- Order your board from Future Electronics
  - **i.MXRT1050-EVKB**
  - **i.MXRT1060-EVK**

What is Edgescale?

Edgescale provides secure device management for IoT edge nodes based on NXP microprocessors and microcontrollers

- OEMs and developers can leverage cloud compute frameworks like AWS Greengrass, Azure IoT and Aliyun on Layerscape and i.MX processors.
- Delivers device security and management needed to securely deploy and manage a large number of Edge computing devices from the cloud.
- Cloud dashboard to securely enroll Edge devices, monitor their health, attest and deploy container applications and firmware updates.

Secure manufacturing support

- Factory provisioning with manufacturing protection
- OTA secure enrollment client
- Code-signing tool

Edge computing device software

- Docker®-based containers
- Secure device management client
- Secure firmware with Arm® TrustZone support
- Secure key/credential management with PKCS.11 API
- Choice of Ubuntu, Yocto, OpenWRT or custom Linux distribution

Secure Device Management Cloud services

- EdgeScale dashboard for users
- EdgeScale CLI for developers and admin
- Secure enrollment service
- Secure device monitoring service
- Secure container deployment service
- Secure firmware management service
- RESTful APIs for customer cloud integration

AI/ML Developer Experience Example – Bring Your Own Model

Bring Your Own Model

Customer brings his/her own ML model to EdgeScale Edge Intelligence service portal

Optimize ML Model

NXP cloud automatically converts and optimizes the model for the target devices

Build and Package

Automatically builds and packages inferencing engine and model for target devices

Deploy to Devices

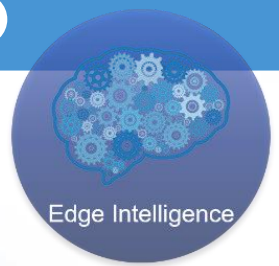
The ML software package is deployed to target devices as OTA update or Docker App

Inferencing at Edge

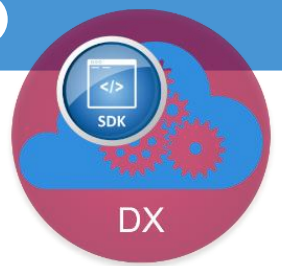
The newly deployed ML app performs ML inferencing on target IoT/Edge devices

STEP 1

STEP 2



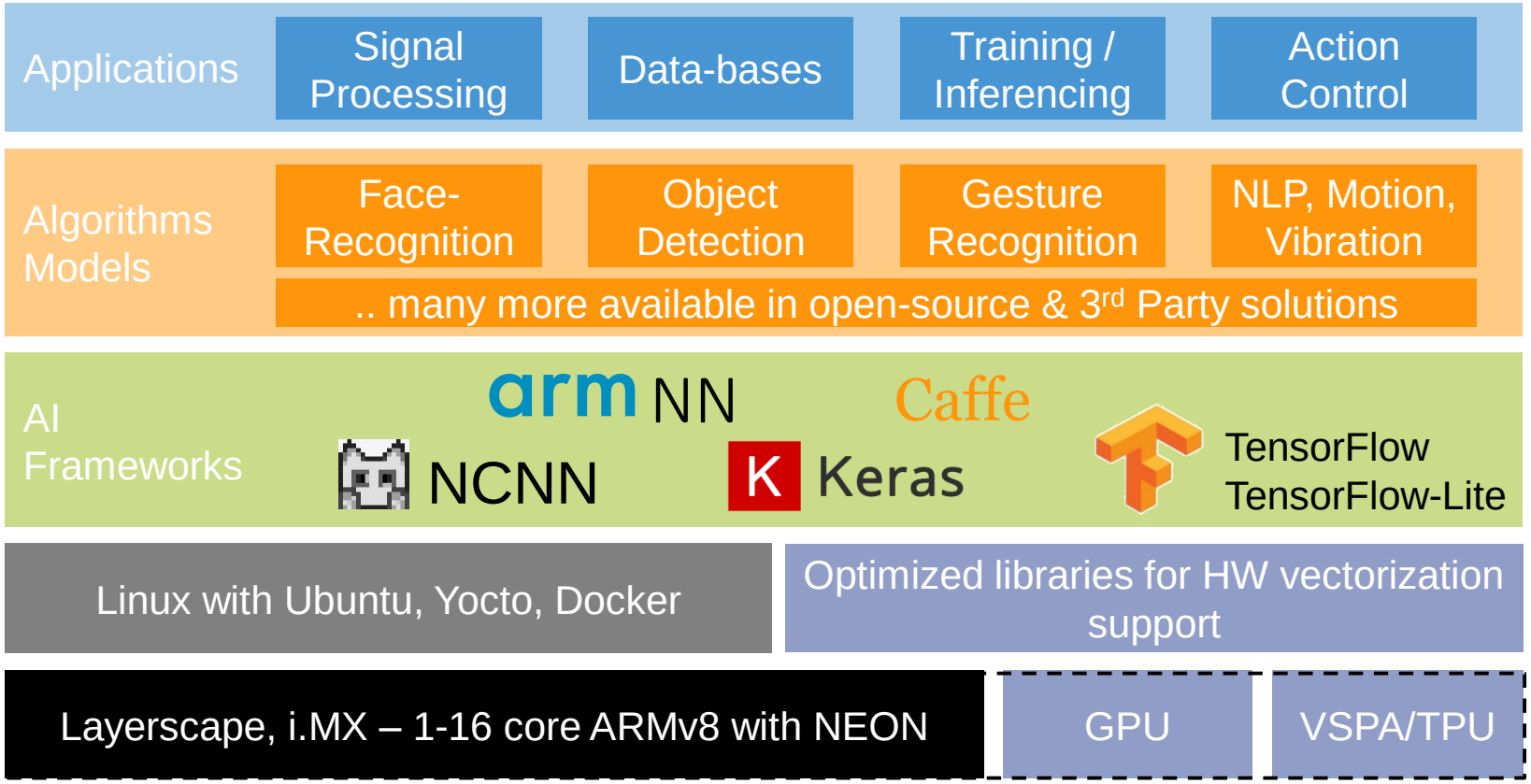
STEP 3

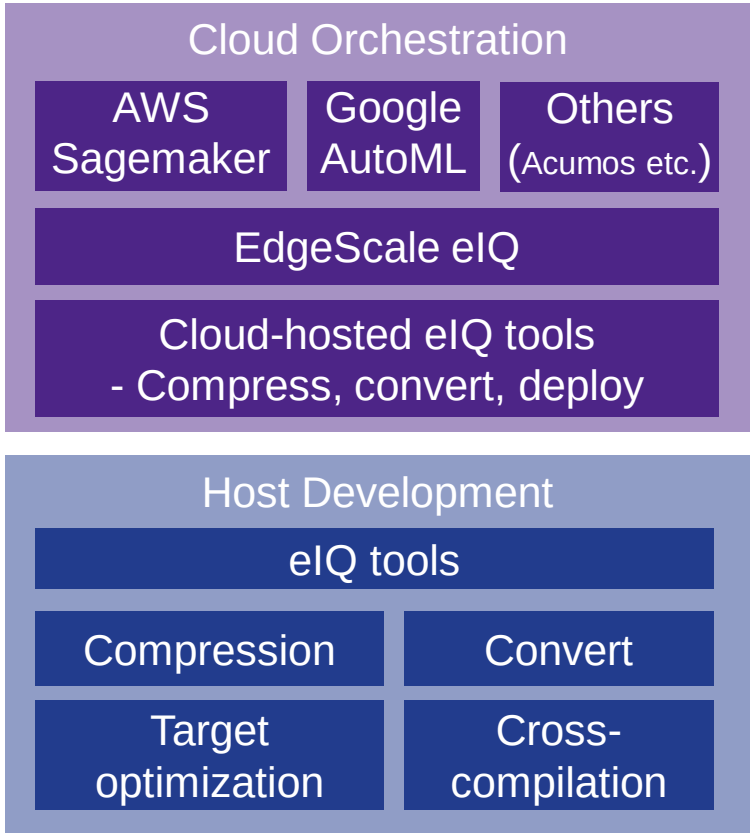


STEP 4

STEP 5

Edgescale and eIQ for AI on Layerscape & i.MX





- NXP provides the right enablement for cloud-connected AI/ML applications @ Edge.
- Host-based eIQ tools for model conversion, optimization and target optimization.
- Edgescale leverages eIQ tools for cloud-based orchestration and integration with Sagemaker, AutoML etc.
- Helps customer leverage open-source frameworks, models and communities.

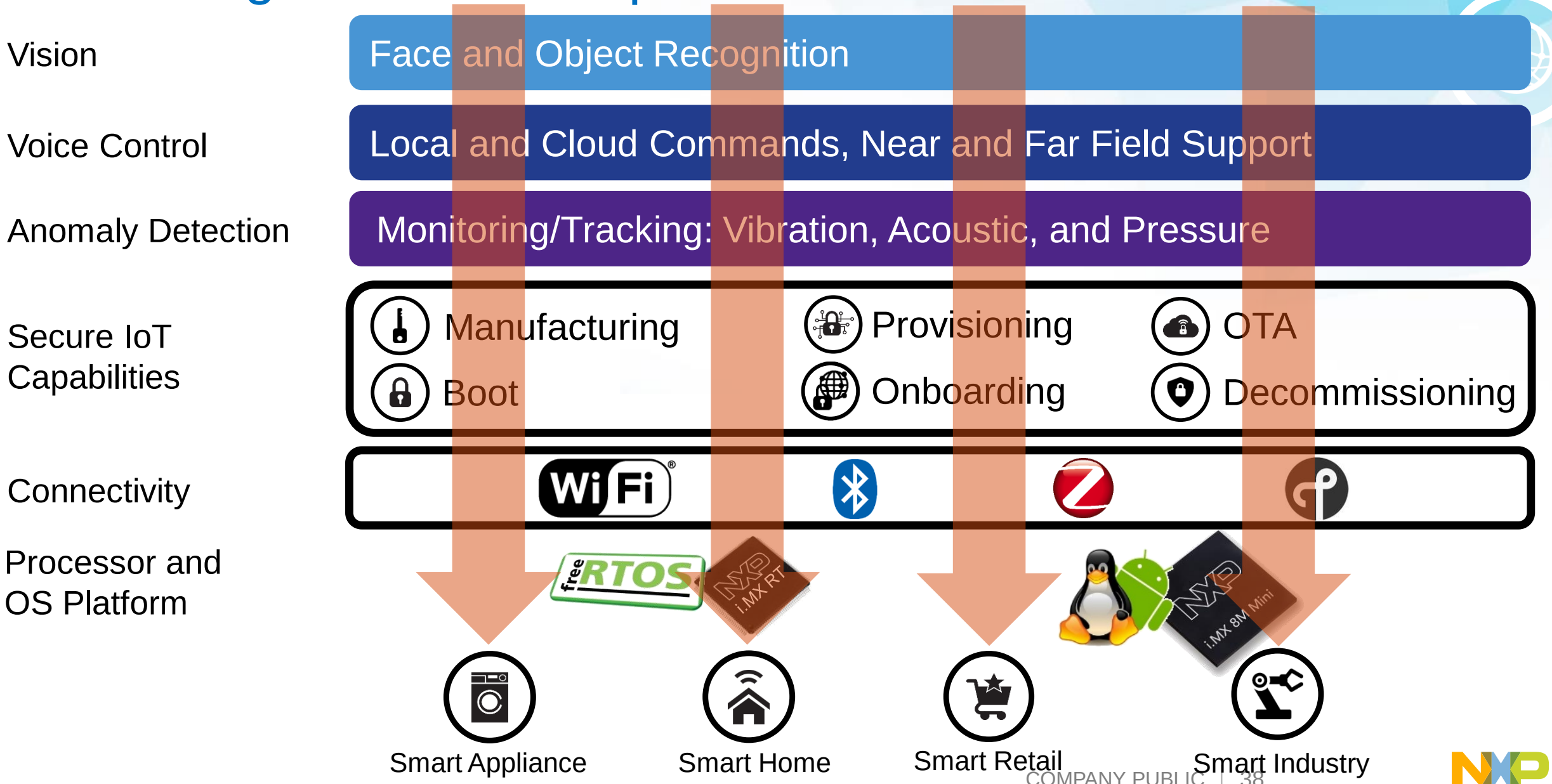
Horizontal Machine Learning Technologies at the IoT Edge

Vision	Face and Object Recognition
Voice Control	Local and Cloud Commands, Near and Far Field Support
Anomaly Detection	Monitoring/Tracking: Vibration, Acoustic, and Pressure

IoT Edge Compute Enabling Technologies

Vision	Face and Object Recognition
Voice Control	Local and Cloud Commands, Near and Far Field Support
Anomaly Detection	Monitoring/Tracking: Vibration, Acoustic, and Pressure
Secure IoT Capabilities	Manufacturing, Provisioning, OTA, Boot, Onboarding, Decommissioning
Connectivity	Wi Fi, Bluetooth, Zigbee, Thread
Processor and OS Platform	 

Combining Horizontal Capabilities to Build Vertical Solutions



Example Customer Engagements Today

Voice Control

Anomaly Detection

Secure Facial Recognition

Empowering Future Products

Smart lock

- Face unlocks
- Local voice commands (after face rec.)

Smart appliance

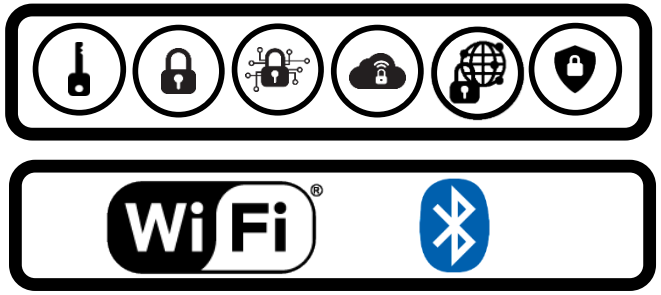
- Anomaly detection for predictive maintenance
- Voice commands “Wash cold, heavily soiled”

Smart appliance / Smart Panel

- Embedded smart display voice assistant, video calling
- Secure facial recognition
- Anomaly detection for predictive maintenance

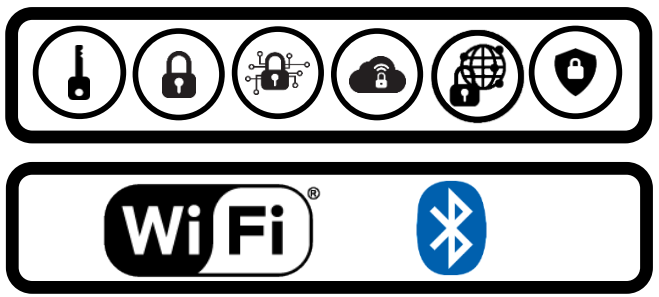
Face Recognition

Voice Control



Anomaly Detection

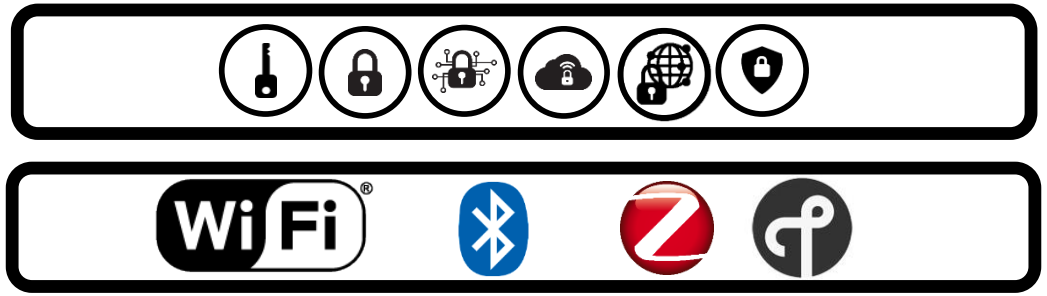
Voice Control



Secure Facial Recognition

Anomaly Detection

Voice Control

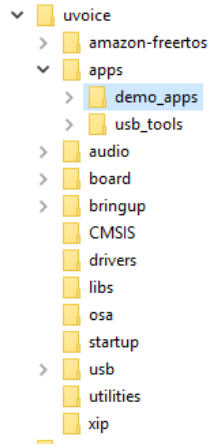


Production Grade, Certified IoT ML Edge Compute Solutions

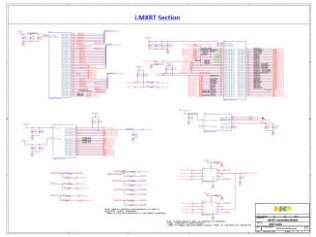
- Implemented with best in class silicon, software and IP from NXP and 3rd parties
- Near production ready hardware
  - Cost and form factor optimized
- Pre-integrated production ready software, fully tested & certified
- NXP provides a single point of contact for support, licensing and procurement
- Use case dependent solutions:
  - Turnkey – for well-defined use cases
  - Customizable – can be modified, tuned and trained for specific use cases

OOB HW/SW

Software Source



Schematics



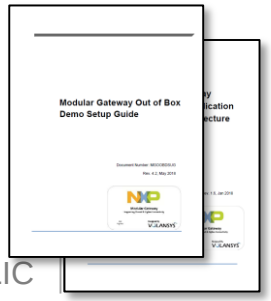
Layouts

Certifications

BOMs

U1	MIMXRT1052DVL6B	i.MXRT1050 Cross Processor
U3	W9812G6JB-6I	128Mbit SDRAM 3V 166MHz
U4	IS26KL256S-DABLI00	256Mb Hyperflash 3V 100MHz Fl
U5	A7101CHTK2	Secure IoT/Authentication Conn
U6	LBEE5KL1DX-883	IC WIFI BT/BLE B/G/N 3-4.8V LGA
U7	MKW21Z512VHT4	KINETIS L 32-BIT MCU CORTEX-M
U8,U9	NX3L2267GM,115	IC ANALOG SWITCH SPDT 10XQFN
U10	XCL214B333DR	DC/DC CONVERTER 3.3V 5W

Documentation



Alexa Voice Service for MCU Solution

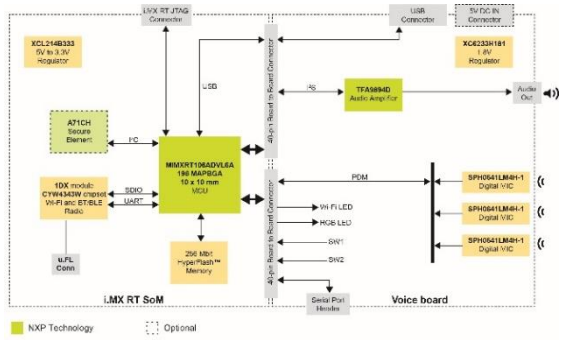
Easily & inexpensively add Alexa built-in to anything – Light switches, smart plugs, smart appliances, smoke detectors, garage door openers, thermostats, etc.

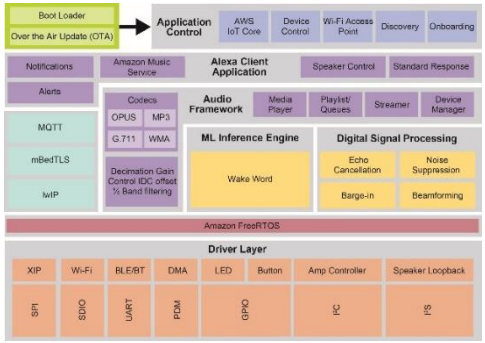
Features

- Brings Alexa to MCU developers for the first time
- Amazon qualified hardware & software
- Audio Front End, Alexa WWE, Alexa for MCU stack
- Over $10 lower BOM cost vs. MPU implementations
- No SRAM, eMMC Flash, or PMIC & fewer PCB layers
- Audio tuning support services available from NXP
- Compact footprint to fit any application design
- Available now (limited access), mid-year (distribution $49)

Key NXP Content

- i.MX RT106A – Arm Cortex-M7 600 MHz (1MB SRAM)
- TFA9894D/N2 – Smart class D amplifier
- A71CH secure element (optional)
- Pre-int. SW inc. Alexa WWE, AVS for MCU stack, AFE (no ext. DSP)





Early access now. In distribution July 2019

Local Voice Commands for MCU Dev. Kit

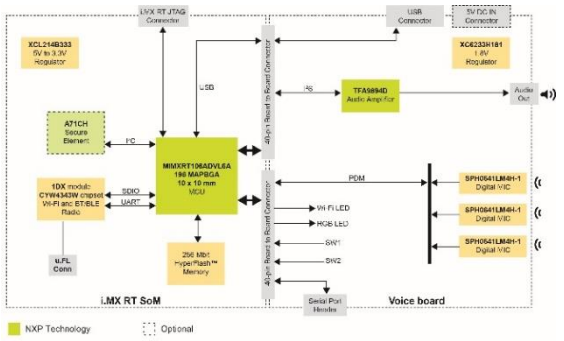
Easily & inexpensively add local voice control to anything, no cloud connection necessary
– Music player, light switch, washer machine, miscellaneous & generic commands

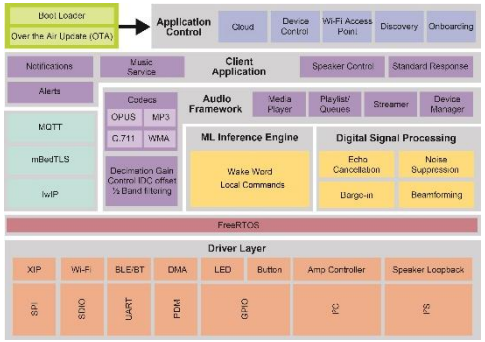
Features

- Production ready hardware & software
- Out of the box local commands development kit
- No third party licenses needed for AFE or ASR engine
- Custom wake word & commands available (NRE)
- Over $10 lower BOM cost vs. MPU implementations
- No SRAM, eMMC Flash, or PMIC and fewer PCB layers
- Audio tuning support services available from NXP
- Compact footprint

Key NXP Content

- i.MX RT106L – Arm Cortex-M7 600 MHz (1MB SRAM)
- TFA9894D/N2 – Smart class D amplifier
- A71CH secure element (optional)
- Pre-integrated SW including local commands & custom wake words





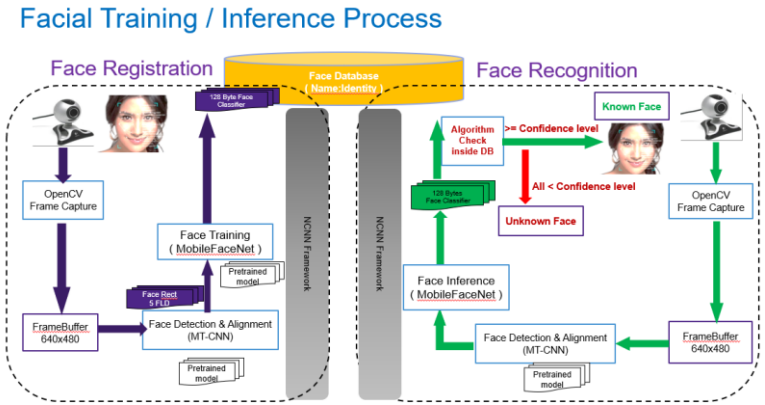
In development, availability 4Q19

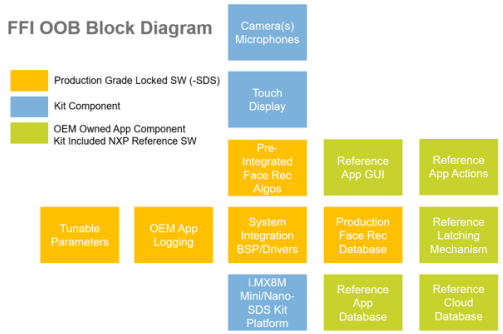
Friction Free Face Recognition Solution

Friction free interface user recognition & personalization – smart appliances, smart thermostats, smart screens

Features

- Automatically registers faces to recognize users (name not required)
- Ability to connect User ID to OEM defined actions
- Production grade HW & SW with example use case app
- Production grade APIs ↔ black box I/F
- Black box production grade face → User ID
- OEM ties to Actions
- Debug/logging facility
- Instantaneous registration
- Frame rates up to 10 fps @ 720p, <200ms inference time
- Low light & no light capable camera with IR cut filter





Key NXP Content

- i.MX 8M Mini
- Quad Arm Cortex-A53 @1500 MHz

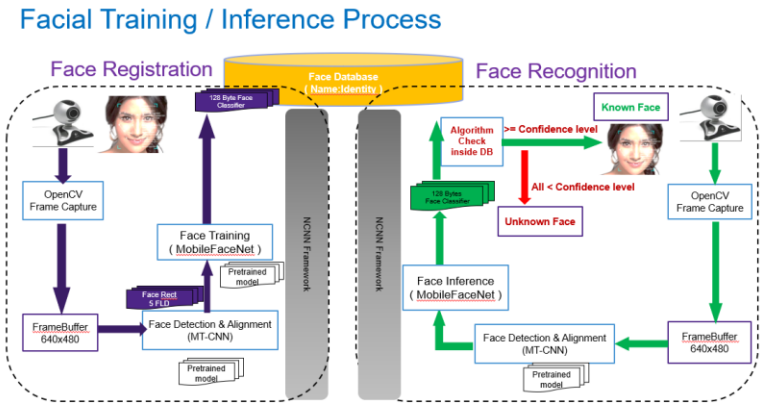
In development, availability 4Q19

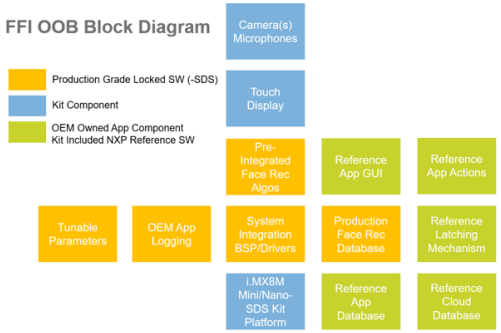
Friction Free Face Recognition MCU Solution

Friction free interface user recognition & personalization – smart appliances, smart thermostats, smart screens

Features

- Automatically registers faces to recognize users (name not required)
- Ability to connect User ID to OEM defined actions
- Production grade HW & SW with example use case app
- Production grade APIs ⟷ black box I/F
- Black box production grade face → User ID
- OEM ties to Actions
- Debug/logging facility
- Quick registration
- <750ms inference time
- 10's of faces





Key NXP Content

- i.MX RT106F
- Arm Cortex-M7 600 MHz (1MB SRAM)

In development, availability 4Q19

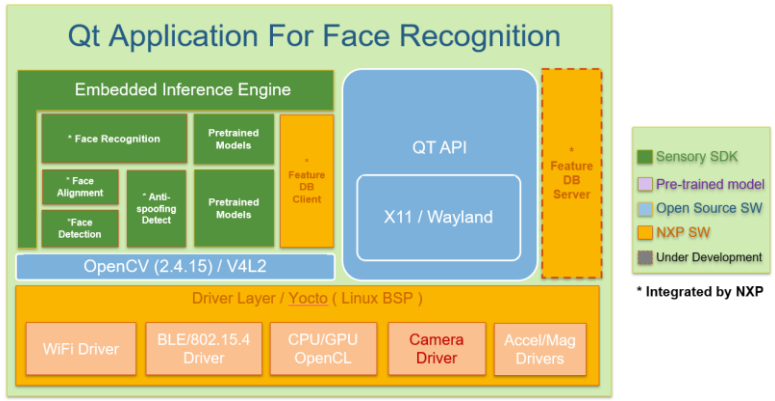
Secure ID Face Recognition Solution

Embed secure user ID enrollment anywhere – smart alarm & access panels, smart appliances, smart thermostats, smart screens

Features

- Robust ID to securely enroll individuals
- Gives OEMs ability to connect securely identified individual to application actions
- OEM OOB Sequence
  - Provides black box production grade secure ID access facility which OEM ties to actions
- OOB Includes
  - Production grade kit
  - Production grade pre-integrated SW
  - Example use case app reference SW
  - Production APIs
  - Debug/Logging facility

Sensory Facial Recognition (Local QT APP) – Software Block Diagram



Key NXP Content

- i.MX 8M Mini
- Quad Arm Cortex-A53 @1500 MHz

In development, availability 4Q19

Early Access NOW, Launch Q4 '19

MCU ML Anomaly Detection Solution

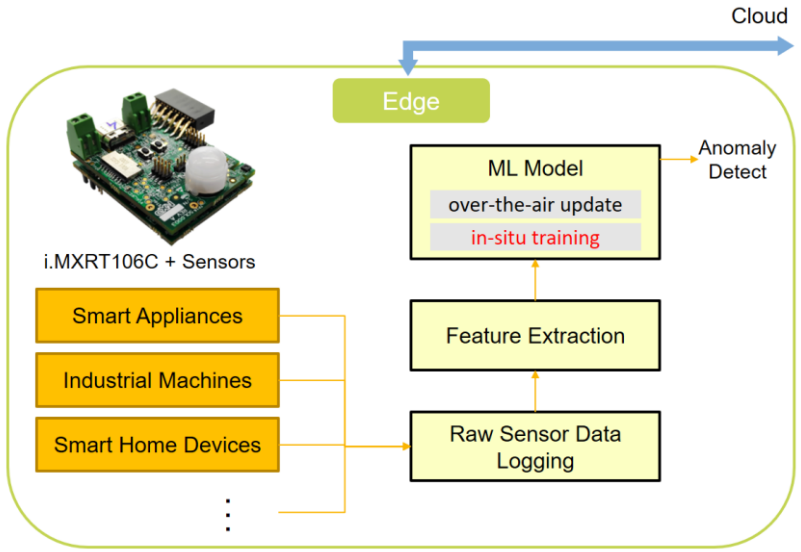
Predictive machine maintenance & lifecycle intelligence for smart appliances and industrial machines, audio event detection. Supports local and cloud learning and monitoring.

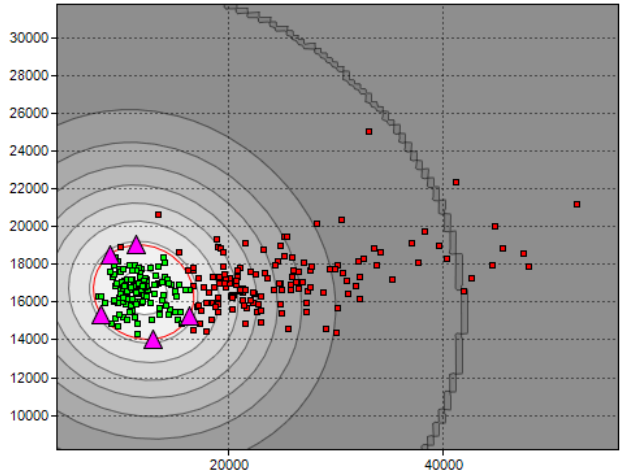
Features

- Environmental awareness & structural monitoring
  - Vibration, sound, pressure, light, EMI, etc
- Acoustic event with pressure and audio anomaly
  - Security system in a box
- ML based anomaly detection
  - Support Vector Machine (SVM)
  - Needs only “good” data, no need to run to failure

Key NXP Content

- i.MX RT106C Arm Cortex-M7 600 MHz (1MB SRAM)
- FXOS8300CQ accelerometer
- NPS3000VV differential pressure sensor





Anomaly Detect

In development, availability 4Q19

Early Access NOW, Launch Q4 ‘19

Summary

- Edge AI, requiring real-time responsiveness is transforming system design and entire industries
- NXP has a broad portfolio of processing solutions, well enabled for the demanding smarter edge
- NXP provides the hardware, reference designs, software tools, and services to enable customers to implement AI today
- Contact us to discuss how we can help you meet your AI goals

Wrap-Up and Q&A

Further Reading

- NXP eIQ
- NXP Edgescale
- TensorFlow Lite
- CMSIS-NN

Machine Learning Courses:

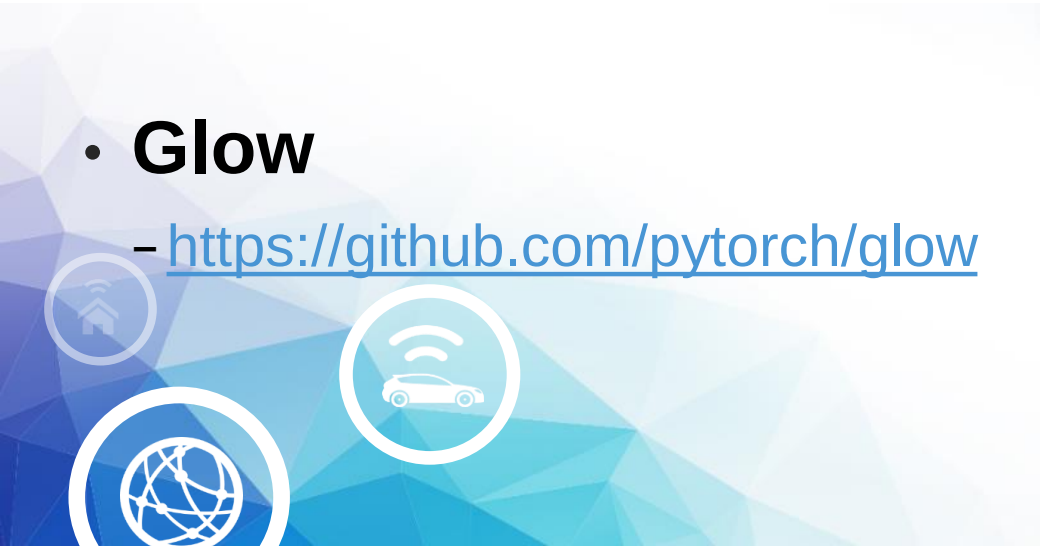
- Video series on Neural Network basics
- Google TensorFlow Lab
- Google Machine Learning Crash Course
- Google Image Classification Practica
- Doulos Deep Learning Course

Git Repos

- **TensorFlow Lite**
  - https://github.com/tensorflow/tensorflow/tree/v1.13.1/tensorflow/lite

- **CMSIS-NN**
  - https://github.com/ARM-software/CMSIS_5/tree/master/CMSIS/NN
  - CIFAR-10: https://github.com/ARM-software/ML-examples/tree/master/cmsisnn-cifar10
  - KWS: https://github.com/ARM-software/ML-KWS-for-MCU

- **Glow**
  - https://github.com/pytorch/glow

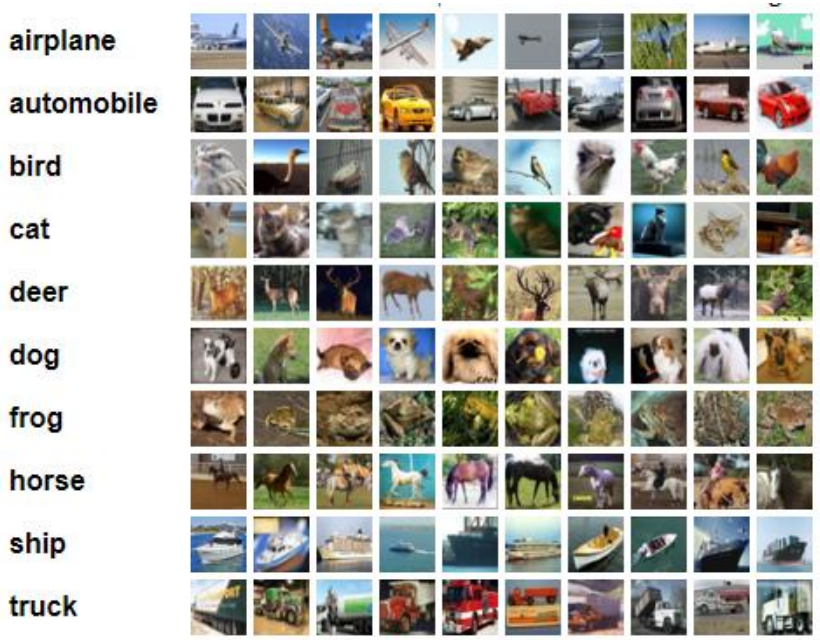


Back-up

eIQ i.MXRT Examples

CIFAR-10

- The “Hello World” version of Machine Learning
- CIFAR-10 demos uses models trained on the CIFAR-10 dataset to classify 32x32 pixel images into 10 different categories.
  - airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks
- Details on CIFAR-10 (pronounced SEE-far) can be found at https://www.cs.toronto.edu/~kriz/cifar.html



CIFAR-10 Input

- Images are 32x32 pixels
- Image is composed of 3072 bytes. The first 1024 bytes are the red channel values, the next 1024 the green, and the final 1024 the blue.
- Use **cifar10_png_to_array.py** script included in eIQ to convert 32x32 PNG image to the proper format.
- Details in eIQ Demo Input Guide

Keyword Spotting (KWS)

- Key Word Spotting uses neural networks to analyze speech and spot specific keywords.

- The model in the CMSIS-NN KWS example can identify 10 different keywords:
  - Yes, No, Up, Down, Left, Right, On, Off, Stop, Go

- The model in the TensorFlow Lite KWS example can identify 2 different keywords:
  - Yes, No

Keyword Spotting (KWS) Input

- In alpha eIQ release is only possible to create new audio snippets that are correctly recognized for CMSIS-NN example
- Uses pre-recorded 16000Hz mono WAV files converted to binary array. Microphone support in development.

- Use the **kws_transform_cmsisnn.py** script found in eIQ release to convert WAV file to proper format.
- Details in eIQ Demo Input Guide

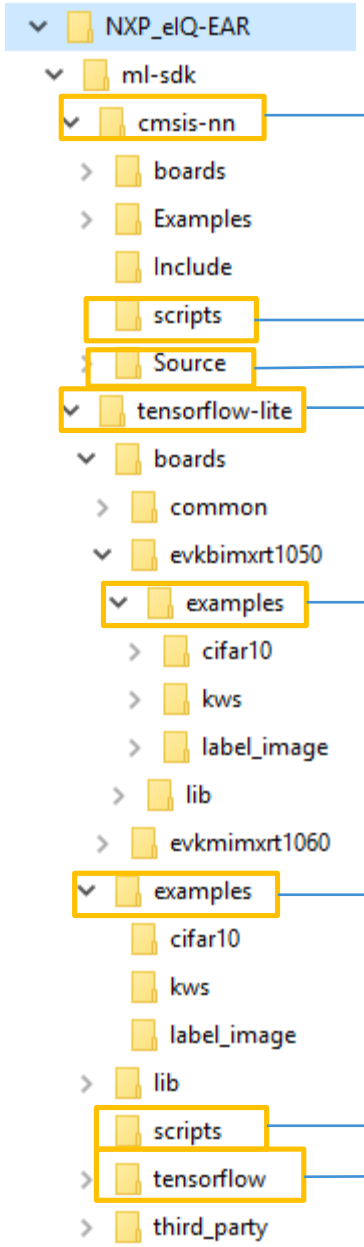
Label Image

- Uses a Mobilenet model to categorize an image into one of 1000 categories.
  - Model is February 2018 release of mobilenet_v1_0.25_128_quant
  - Model trained on images from ImageNet
- Only TensorFlow Lite example available.
- More advanced image categorization than the CIFAR-10 example.

Label Image Input

- Uses a 24-bit bitmap (BMP) image that has been converted to a C array. Camera support is under development.
- Any image larger than 128x128 that is in one of the 1000 categories listed in \tensorflow-lite-imx-rt\examples\label_image\labels.h should work
  - However model has 39.5% Top-1 accuracy and 64.4% Top-5 accuracy so be aware that new image input may not be properly categorized.
- Details in eIQ Demo Input Guide

eIQ Folder Structure



CMSIS-NN Folder

CMSIS-NN Example Data Scripts

CMSIS-NN Source Code

TensorFlow Lite Folder

Project Files for Examples

Source Code for Examples

TensorFlow Example Data Scripts

TensorFlow Lite Source Code

TensorFlow Lite “Label Image” Example Walkthrough

- Written in C++. Support for MCUXpresso IDE and IAR

- Include image, model, and labels

```
34 #include "stopwatch_image.h"                    //Image to analyze
35 #include "mobilenet_v1_0.25_128_quant_model.h"  //Model
36 #include "labels.h"                             //Categories to label image
37
```

- Load tflite converted model with BuildFromBuffer()

```
67 void RunInference(Settings* s) {
68   std::unique_ptr<tflite::FlatBufferModel> model;
69   std::unique_ptr<tflite::Interpreter> interpreter;
70   model = tflite::FlatBufferModel::BuildFromBuffer(mobilenet_model, mobilenet_model_len);  //Load model
```

- Load image and set to input tensor

```
105   uint8_t* in = read_bmp(stopwatch_bmp, stopwatch_bmp_len, &image_width, &image_height,
106                          &image_channels, s);
107
108   int input = interpreter->inputs()[0];
```

```
135       resize<float>(interpreter->typed_tensor<float>(input), in, image_height,
136                     image_width, image_channels, wanted_height, wanted_width,
137                     wanted_channels, s);
```

TensorFlow Lite “Label Image” Example Walkthrough

- Run inference with Invoke()

```
150   auto start_time = GetTimeInUS();
151   for (int i = 0; i < s->loop_count; i++) {
152     if (interpreter->Invoke() != kTfLiteOk) {
153       LOG(FATAL) << "Failed to invoke tflite!\r\n";
154     }
155   }
156   auto end_time = GetTimeInUS();
157   LOG(INFO) << "Average time: " << (end_time - start_time) / 1000 << " ms\r\n";
158
```

- Get results from output tensor

```
163   int output = interpreter->outputs()[0];
164   TfLiteIntArray* output_dims = interpreter->tensor(output)->dims;
165   /* Assume output dims to be something like (1, 1, ... , size) */
166   auto output_size = output_dims->data[output_dims->size - 1];
167   switch (interpreter->tensor(output)->type) {
168     case kTfLiteFloat32:
169       get_top_n<float>(interpreter->typed_output_tensor<float>(0), output_size,
170                        s->number_of_results, threshold, &top_results, true);
```

```
186   if (ReadLabels(labels_txt, &labels, &label_count) != kTfLiteOk)
187     return;
188
189   LOG(INFO) << "Detected:\r\n";
190   for (const auto& result : top_results) {
191     const float confidence = result.first;
192     const int index = result.second;
193     LOG(INFO) << "   " << labels[index] << " (" << (int)(confidence * 100) << "% confidence)\r\n";
```

CMSIS-NN “CIFAR-10” Example Walkthrough

- Written in C
- Include image data, weights, and parameters for model

```
48 #include "inputs.h"       //Image data
49 #include "parameter.h"    //Parameters for model
50 #include "weights.h"      //Weights for model
```

- Load image data

```
151     uint8_t image_data[32 * 32 * 3] = SHIP_IMG_DATA;
```

CMSIS-NN “CIFAR-10” Example Walkthrough

- Call CMSIS-NN APIs to execute model layers

```
98      /* conv1 img_buffer2 -> img_buffer1 */
99      arm_convolve_HWC_q7_RGB(img_buffer2, CONV1_IM_DIM, CONV1_IM_CH, conv1_wt, CONV1_OUT_CH, CONV1_KER_DIM, CONV1_PADDING,
100                             CONV1_STRIDE, conv1_bias, CONV1_BIAS_LSHIFT, CONV1_OUT_RSHIFT, img_buffer1, CONV1_OUT_DIM,
101                             (q15_t *) col_buffer, NULL);
102
103     arm_relu_q7(img_buffer1, CONV1_OUT_DIM * CONV1_OUT_DIM * CONV1_OUT_CH);
104
105     /* pool1 img_buffer1 -> img_buffer2 */
106     arm_maxpool_q7_HWC(img_buffer1, CONV1_OUT_DIM, CONV1_OUT_CH, POOL1_KER_DIM,
107                        POOL1_PADDING, POOL1_STRIDE, POOL1_OUT_DIM, NULL, img_buffer2);
108
109     /* conv2 img_buffer2 -> img_buffer1 */
110     arm_convolve_HWC_q7_fast(img_buffer2, CONV2_IM_DIM, CONV2_IM_CH, conv2_wt, CONV2_OUT_CH, CONV2_KER_DIM,
111                              CONV2_PADDING, CONV2_STRIDE, conv2_bias, CONV2_BIAS_LSHIFT, CONV2_OUT_RSHIFT, img_buffer1,
112                              CONV2_OUT_DIM, (q15_t *) col_buffer, NULL);
113
114     arm_relu_q7(img_buffer1, CONV2_OUT_DIM * CONV2_OUT_DIM * CONV2_OUT_CH);
115
116     /* pool2 img_buffer1 -> img_buffer2 */
117     arm_maxpool_q7_HWC(img_buffer1, CONV2_OUT_DIM, CONV2_OUT_CH, POOL2_KER_DIM,
118                        POOL2_PADDING, POOL2_STRIDE, POOL2_OUT_DIM, col_buffer, img_buffer2);
119
120     /* conv3 img_buffer2 -> img_buffer1 */
121     arm_convolve_HWC_q7_fast(img_buffer2, CONV3_IM_DIM, CONV3_IM_CH, conv3_wt, CONV3_OUT_CH, CONV3_KER_DIM,
122                              CONV3_PADDING, CONV3_STRIDE, conv3_bias, CONV3_BIAS_LSHIFT, CONV3_OUT_RSHIFT, img_buffer1,
123                              CONV3_OUT_DIM, (q15_t *) col_buffer, NULL);
124
125     arm_relu_q7(img_buffer1, CONV3_OUT_DIM * CONV3_OUT_DIM * CONV3_OUT_CH);
126
127     /* pool3 img_buffer-> img_buffer2 */
128     arm_maxpool_q7_HWC(img_buffer1, CONV3_OUT_DIM, CONV3_OUT_CH, POOL3_KER_DIM,
129                        POOL3_PADDING, POOL3_STRIDE, POOL3_OUT_DIM, col_buffer, img_buffer2);
130
131     arm_fully_connected_q7_opt(img_buffer2, ip1_wt, IP1_DIM, IP1_OUT, IP1_BIAS_LSHIFT, IP1_OUT_RSHIFT, ip1_bias,
132                                output_data, (q15_t *) img_buffer1);
133
134     arm_softmax_q7(output_data, 10, output_data);
```

CMSIS-NN “CIFAR-10” Example Walkthrough

- Get Results

```
161     /* Get the object class with the highest confidence value */
162     arm_max_q7(output_data, 10, &max_value, &max_index);
163     PRINTF("Predicted class: %s \r\n", labels[max_index]);
```

Benchmarking

- Benchmarking ongoing and optimizations still under development
- Inference time heavily dependent on the particular model
  - Input data does not affect inference time
- Each eIQ example reports inference time

- CIFAR-10 example in IAR with Release compile settings
  - CMSIS-NN: 23ms
  - TensorFlow Lite: 84ms

SECURE CONNECTIONS
FOR A SMARTER WORLD

www.nxp.com