IMPLEMENT YOUR AI/ML APPLICATION AT THE EDGE WITH NXP EIQ™ SOFTWARE DEVELOPMENT ENVIRONMENT

OCTOBER 2019

MASSIMO INCERTI

MILANO NXP TECH DAY

NXP PLUG&TRUST

SECURE CONNECTIONS FOR A SMARTER WORLD

Machine Learning Revolution

Limitless Possibilities

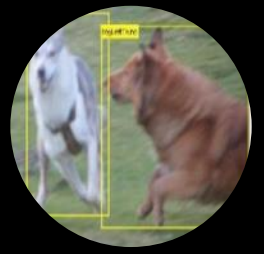
Home Environment

Voice Processing

Gesture Control

Smart Sense & Control

Multi-camera Observation



Personal / Property

Active Object Recognition

Augmented Reality

NXP Enabling Machine Learning Revolution

Training in the cloud

TensorFlow Caffe2

Trained Models
Inference Engines

ONNX Keras

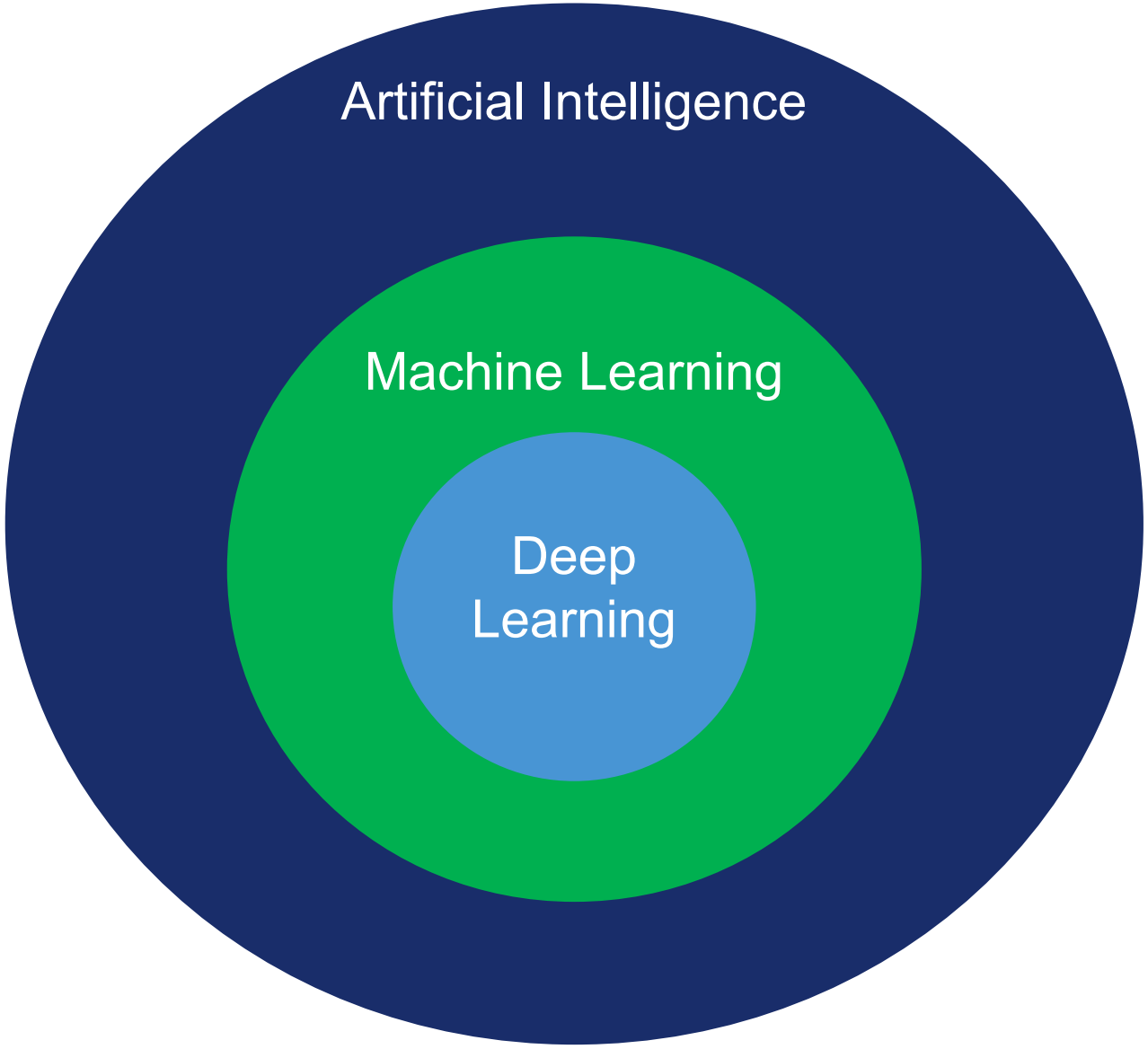
and others

Low-end Edge Compute	Mid-end Edge Compute	High-end Edge Compute
i.MX 6 1-2x Arm 32-bit Crossover & High. Perf. MCUs Arm 32-bit MCU +ML DSP	i.MX 7, 8 1-4x Arm 32/64-bit Performance GPUs Integrated DSP	i.MX 8, Layerscape Multi-core Arm 64-bit High performance • GPUs • DSP • Vector processing

Scalable & optimized inference engines across Embedded Processing continuum

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Artificial Intelligence, Machine Learning, and Deep Learning



Artificial Intelligence

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

Machine Learning

- One of many ways to implement AI
- The concept that if you give machines a lot of data, they can learn how to do smart things on their own without having to be explicitly programmed to do that action.
- Self learning and self improving

Deep Learning

- One of many ways to implement machine learning
- Uses Neural Networks that can learn and make intelligent decisions on its own
- Needs a **lot** of data

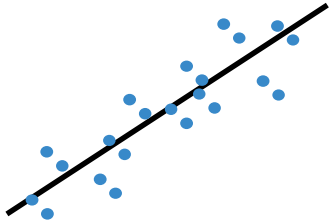
First Stage Considerations for ML at the Edge

- IoT, Industrial, Automotive Application - Can I utilize machine learning?
- Training Time and amount and type of data required for training
- Availability of labeled data (e.g. supervised versus unsupervised)
- Tolerated accuracy
- Number of features
- Computational resources available (e.g. RAM & CPU)
- Latency required/tolerated (cost versus performance)
- Ease of Interpretation
- How will I deploy

What can machine learning do

Regression (Calculation)

- Predict continuous values



X=a, y=?

Classification (Choice)

- Recognition, object detection

It is a ( )
A: Dog B: Cat C: Cow *D: Neither*

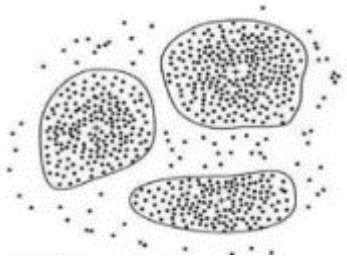
Anomaly detection (Judgement)

- Detect abnormal conditions

Heart is going to malfunction? Y/N

Clustering

- Discover patterns / partitions



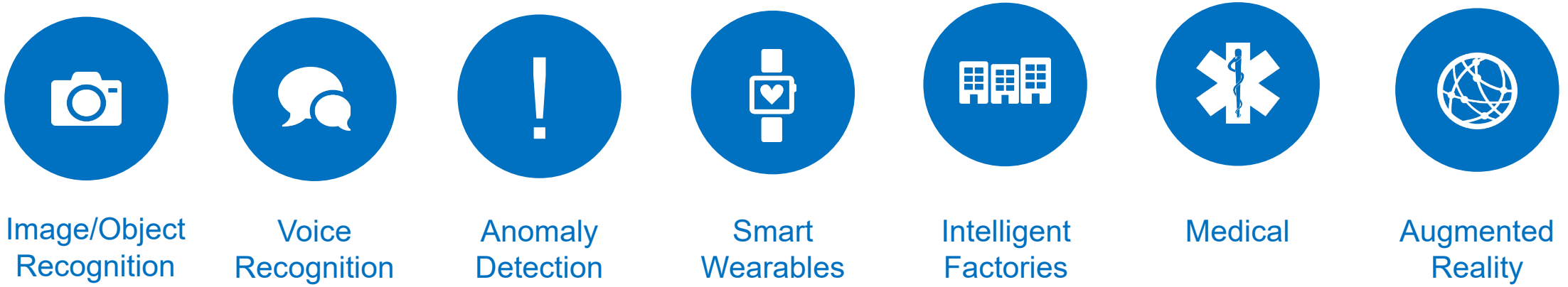
Find crowds
No need labels

Learn strategies

- Reinforcement Learning

How to play the game?

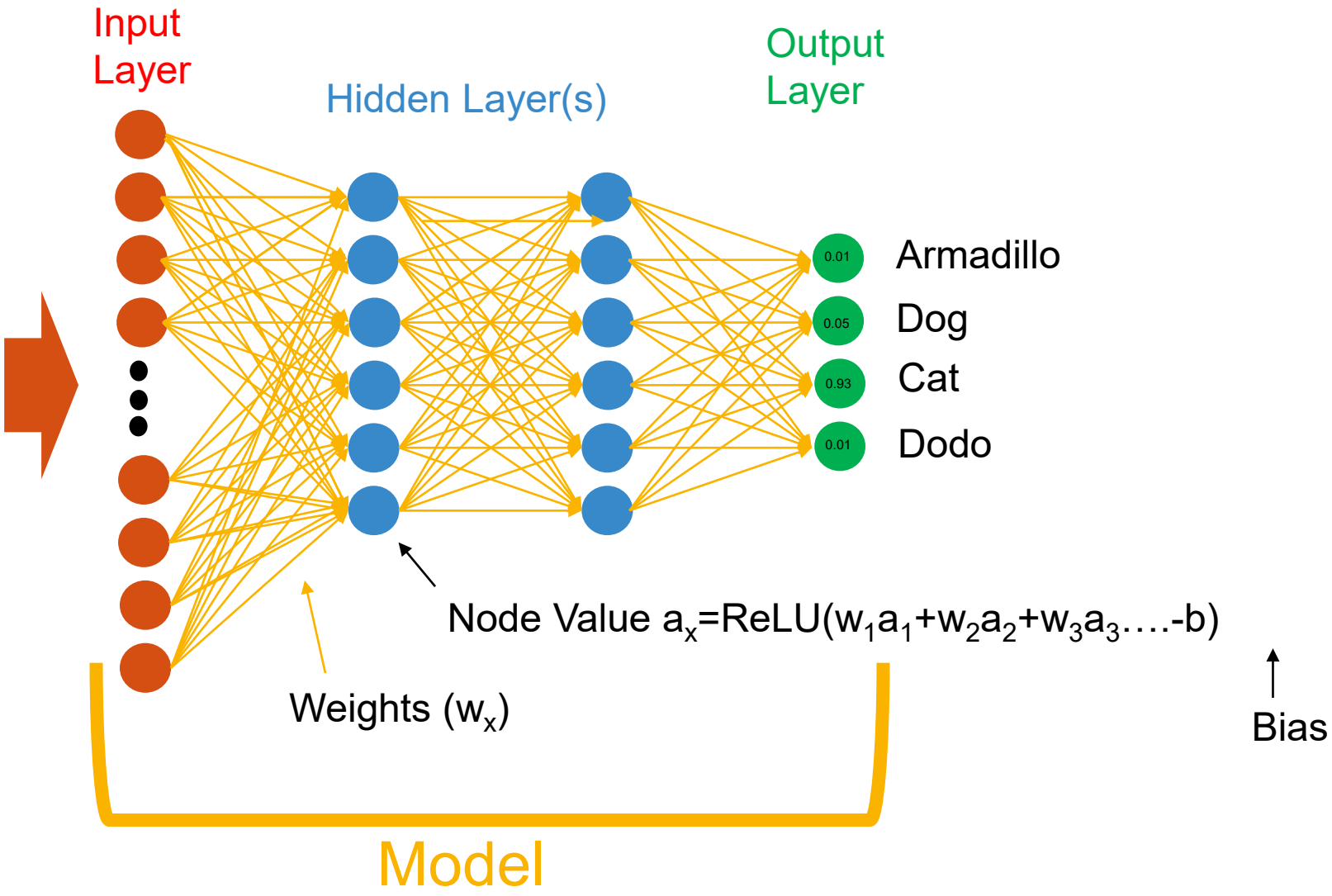
Embedded Machine Learning Applications



Machine Learning Models

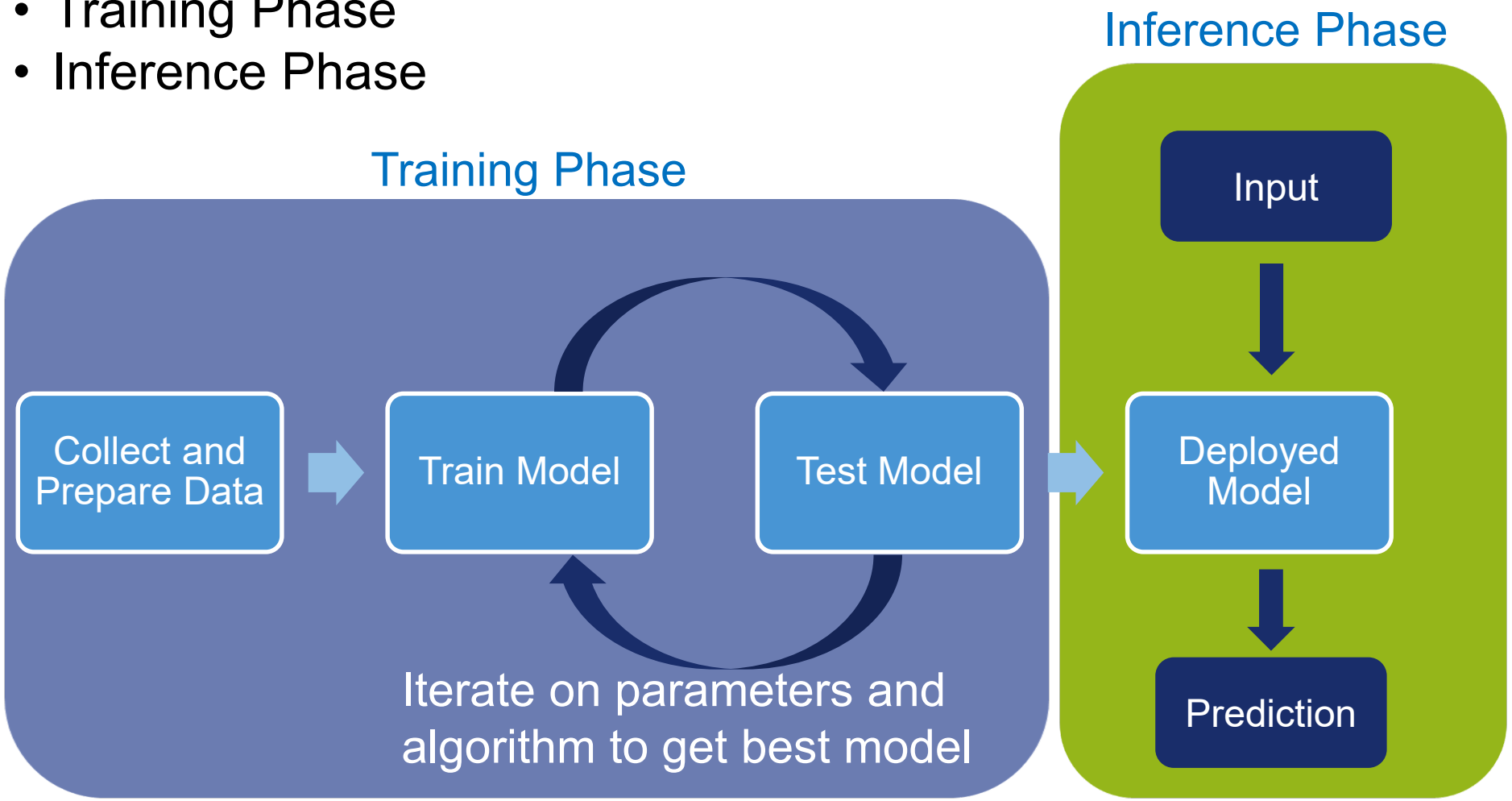
- Models are a mathematical representation of a real-world process
  - ie image recognition, speech recognition, etc
- Essentially a model is a extremely complicated math function that gives a “smart” output value for a given input

Very Simplified Neural Network Model



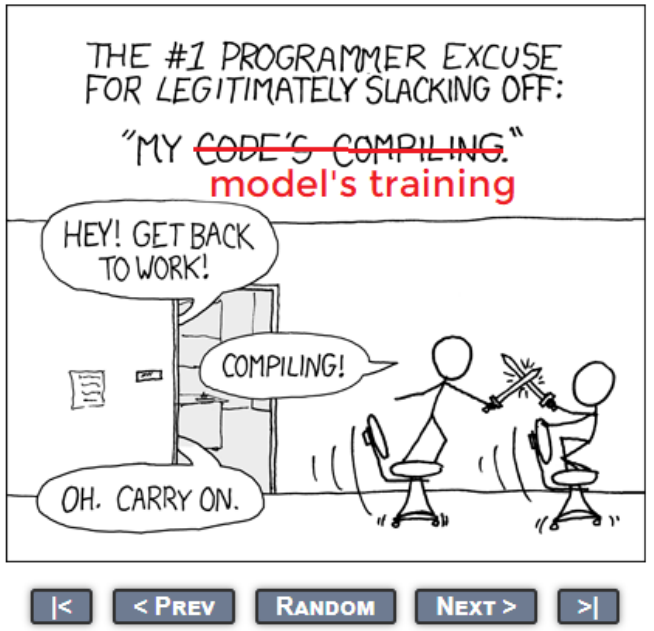
Machine Learning Process

- Training Phase
- Inference Phase



Training Phase

- Training a model is very compute and time intensive
- Involves trying many different weights and biases until get acceptable results over the entire training data
- Difficult to determine what the optimal values are
- Training usually done on CPUs, GPUs, or on cloud
- Due to randomness in the values that are tried, this can result in slightly different weights and biases even if training data is the same
- Could use data taken from the “edge” and upload into cloud for training



Inference Phase

- Inference is using a model to perform evaluations on new data
- Inference time depends on framework and model
- Two possibilities (using image detect as an example):
  1) Upload an image to cloud and evaluate on cloud platform
     - Requires network bandwidth.
     - Latency issues
     - Cloud compute costs
  2) Evaluate image on embedded system itself: Edge Computing
     - Faster response time and throughput
     - Lower Power
     - Don’t need internet connectivity
     - Increased privacy and security

How Are Models Designed? – Model Frameworks

- A framework provides proven APIs and utilities to design, analyze, train, test, validate and deploy models.
- Each framework has their own APIs and methodologies
- Allows developers to focus on overall logic of model, instead of the details of how to implement algorithms or link layers together

```
with tf.variable_scope('conv1') as scope:
  kernel = _variable_with_weight_decay('weights',
                                       shape=[5, 5, 3, 64],
                                       stddev=5e-2,
                                       wd=None)
  conv = tf.nn.conv2d(images, kernel, [1, 1, 1, 1], padding='SAME')
  biases = _variable_on_cpu('biases', [64], tf.constant_initializer(0.0))
  pre_activation = tf.nn.bias_add(conv, biases)
  conv1 = tf.nn.relu(pre_activation, name=scope.name)
  _activation_summary(conv1)

# pool1
pool1 = tf.nn.max_pool(conv1, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1],
                       padding='SAME', name='pool1')
# norm1
norm1 = tf.nn.lrn(pool1, 4, bias=1.0, alpha=0.001 / 9.0, beta=0.75,
                  name='norm1')

# conv2
with tf.variable_scope('conv2') as scope:
  kernel = _variable_with_weight_decay('weights',
                                       shape=[5, 5, 64, 64],
                                       stddev=5e-2,
                                       wd=None)
  conv = tf.nn.conv2d(norm1, kernel, [1, 1, 1, 1], padding='SAME')
  biases = _variable_on_cpu('biases', [64], tf.constant_initializer(0.1))
  pre_activation = tf.nn.bias_add(conv, biases)
  conv2 = tf.nn.relu(pre_activation, name=scope.name)
  _activation_summary(conv2)
```

CIFAR-10 Model in TensorFlow Framework

Model Frameworks

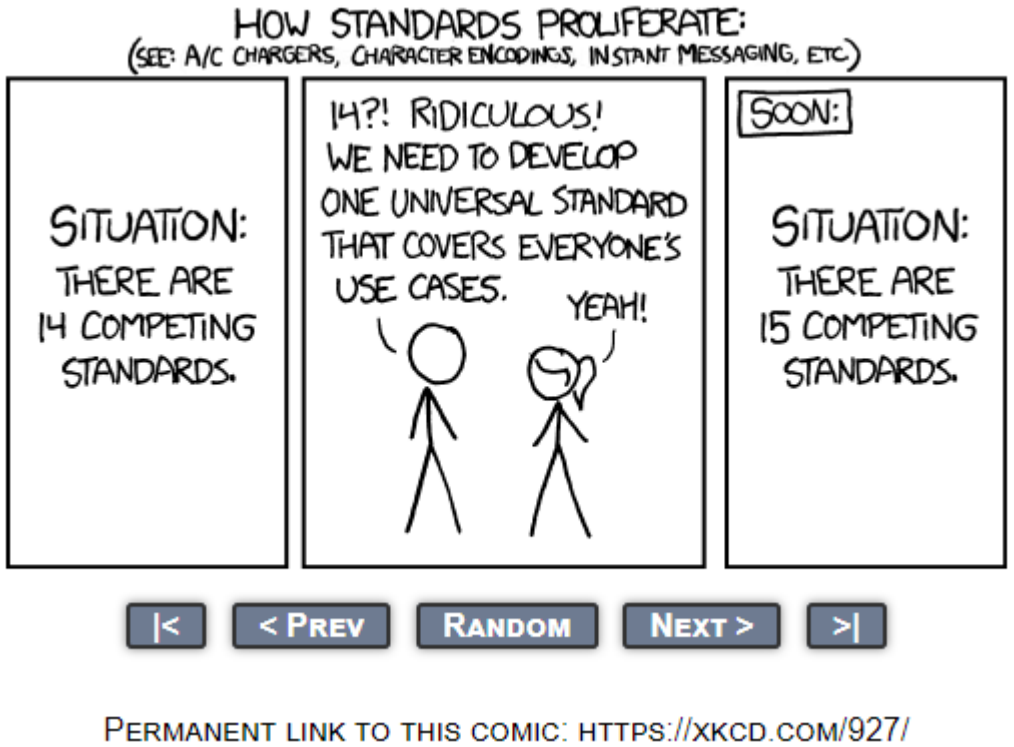
- There are several popular model frameworks in use today.
- This is a constantly changing list as new software is released:
  - TensorFlow – Google framework
  - Keras – higher level API, usually built on top of TensorFlow
  - Caffe2 – Facebook framework
  - PyTorch – Facebook framework

- Python is used for most ML frameworks (interact, build, and train)
- New breakthroughs constantly and favorite framework du jour can change quickly

Universal Framework Translators

There are also universal framework formats like ONNX and NNEF

- Convert different frameworks into a “standard” format



MACHINE LEARNING ACCURACY

Model Accuracy

- Every time an inference is run, most models will give a confidence percentage.
  - This is why last (or one of last) layers in a model is often “softmax” which converts final result to be between 0 and 1.
- Every model has limitations.
- Goal of Machine Learning:
  + Accuracy
  - Size
  - Inference time

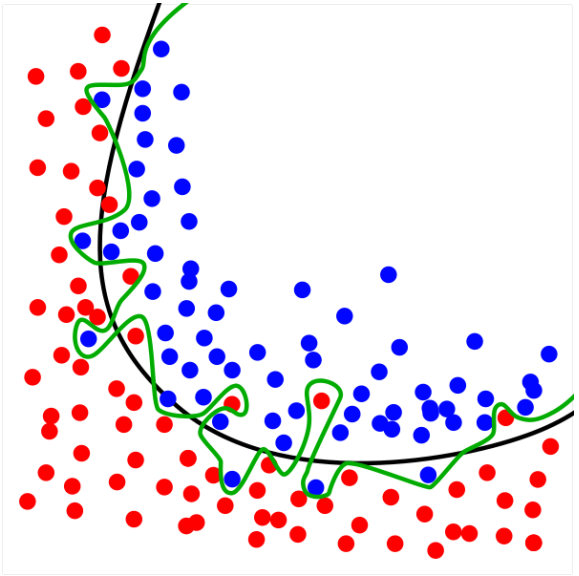
Model Accuracy Continued

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
  - ie: a white horse is not classified correctly if no horses in training data were white
- 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



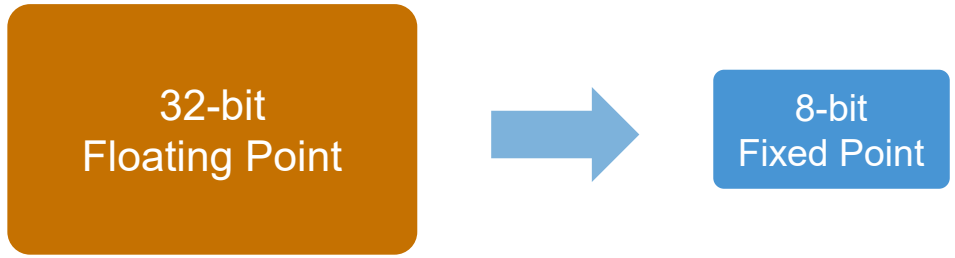
Things That Affect Model Accuracy

- Quality of input training data
- Quantity of input training data
- Model Structure and Training Method
- Efficiency of model conversion for running on embedded system
  - Quantization and Pruning
- Quality of input test data

Quantization and Pruning

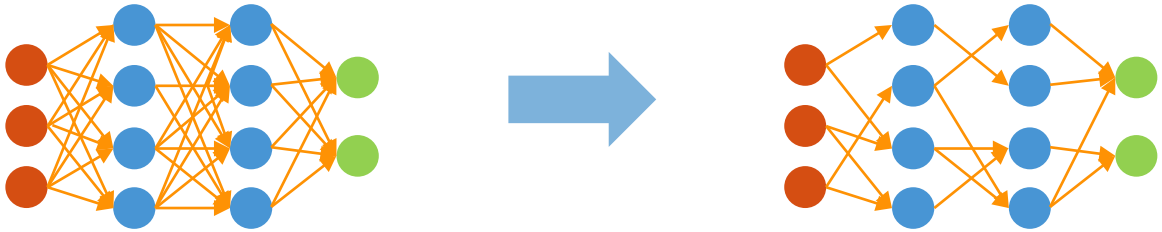
Quantization

- Transform 32-bit floating point weights → 8-bit fixed point weights
  - Reduces model size by 4x
  - Fixed point math quicker than floating point
  - Usually little loss of accuracy



Pruning

- Remove low importance weights and biases from a neural network
  - Recommended to retrain model after pruning



NXP ENABLEMENT ON MACHINE LEARNING

NXP Broad-based ML Solutions (Available Today!)

eIQ™ ML Enablement

- eIQ (edge intelligence) for general-purpose edge AI/ML inference enablement
- **i.MX** 8 family, **i.MX RT**1050/1060

DIY

Third Party SW and HW

- Coral Dev Board
- **i.MX 8M** Development Kit for Amazon® Alexa Voice Service w/ DSP Concepts
- Au-Zone Development Tools

Partners

EdgeScale™ Solution

- Secure deployment of applications (incl. AI/ML) through **docker** containers
- Layerscape devices now; adding i.MX

Think Docker

Turnkey Solutions

- AVS Solution - i.MX RT106A
- Coming soon:
  Anomaly detection and facial recognition solutions based on i.MX RT, i.MX 8M Mini

Fully Tested

EIQ

eIQ - Edge Intelligence

Collection of Libraries and Development Tools for Building Machine Learning Apps Targeting NXP MCUs and App Processors

Deploying open-source inference engines

Integration and optimization of neural net (NN) inference engines (Arm NN, Arm CMSIS-NN, OpenCV, TFLite, ONNX, etc.)

End-to-end examples demonstrating customer use-cases (e.g. camera → inference engine)

Support for emerging neural net compilers (e.g. Glow)

Suite of classical ML algorithms such as support vector machine (SVM) and random forest

Integrated into Yocto Linux BSP and MCUXpresso SDK

No separate SDK or release to download

- iMX: New layer meta-imx-machinelearning in Yocto
- MCU: Integrated in MCUXpresso SDK middleware

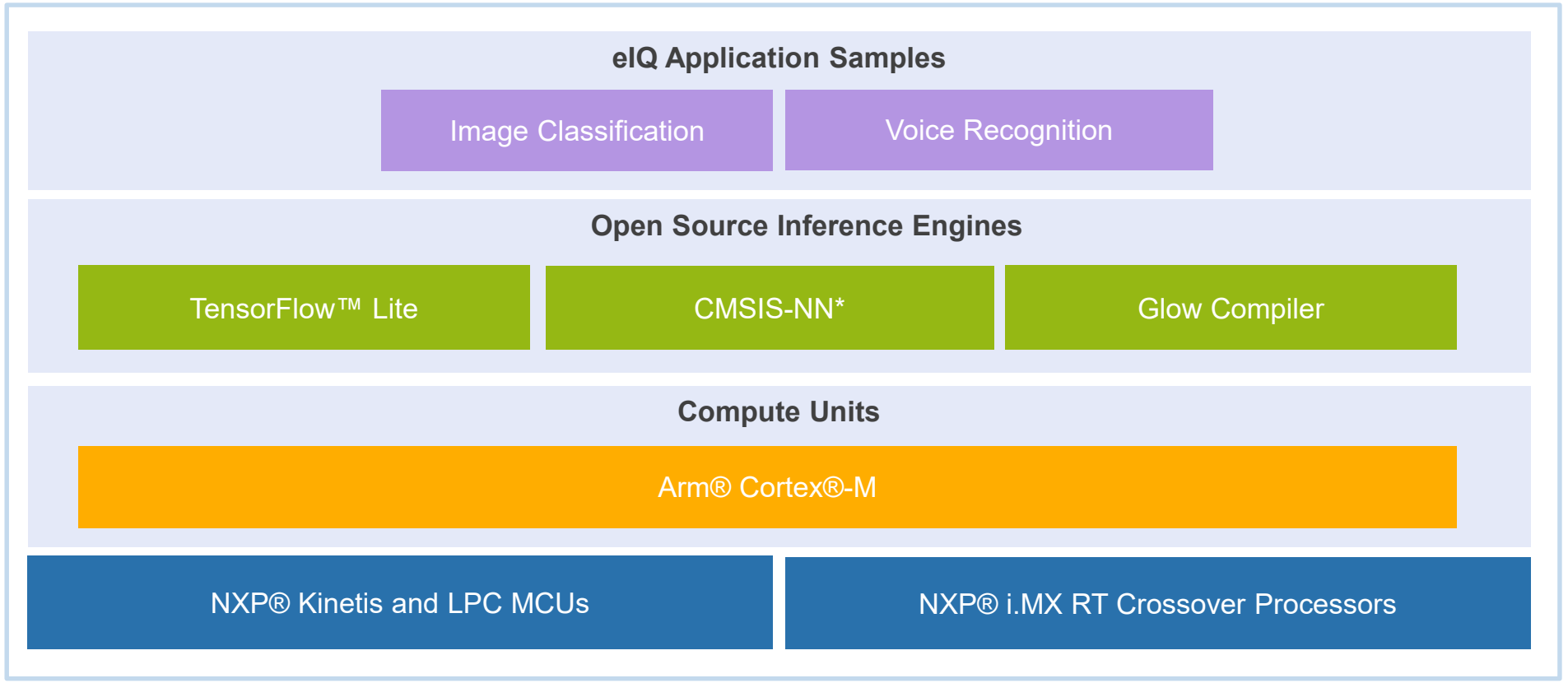
Supporting materials for ease of use

Documentation: eIQ White Paper, Release Notes, eIQ User's Guide, Demo User's Guide

Guidelines for importing pretrained models based on popular NN frameworks (e.g. TensorFlow, Caffe)

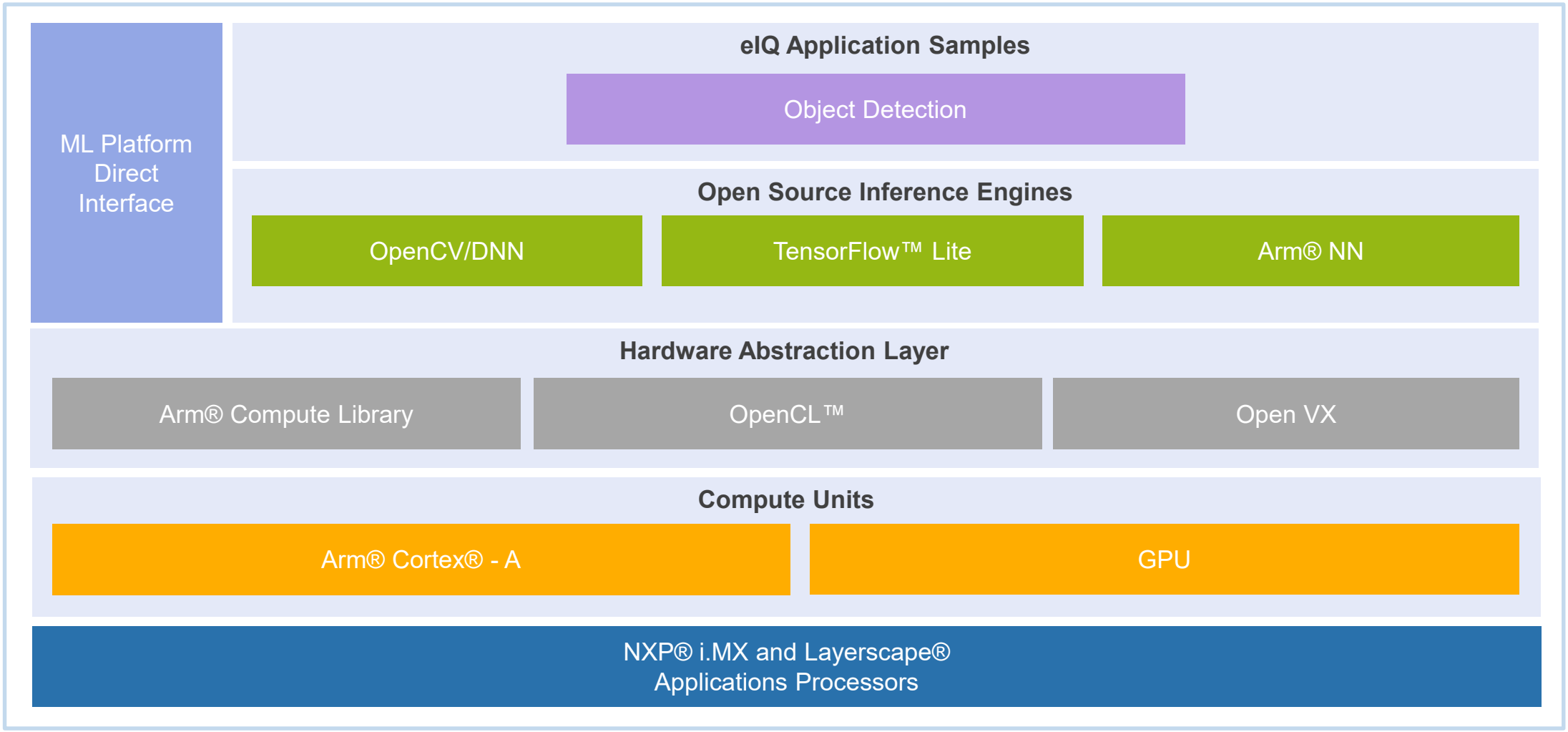
Training collateral for CAS, DFAEs and customers (e.g. lectures, hands-on, video)

eIQ Component Diagram for MCUs



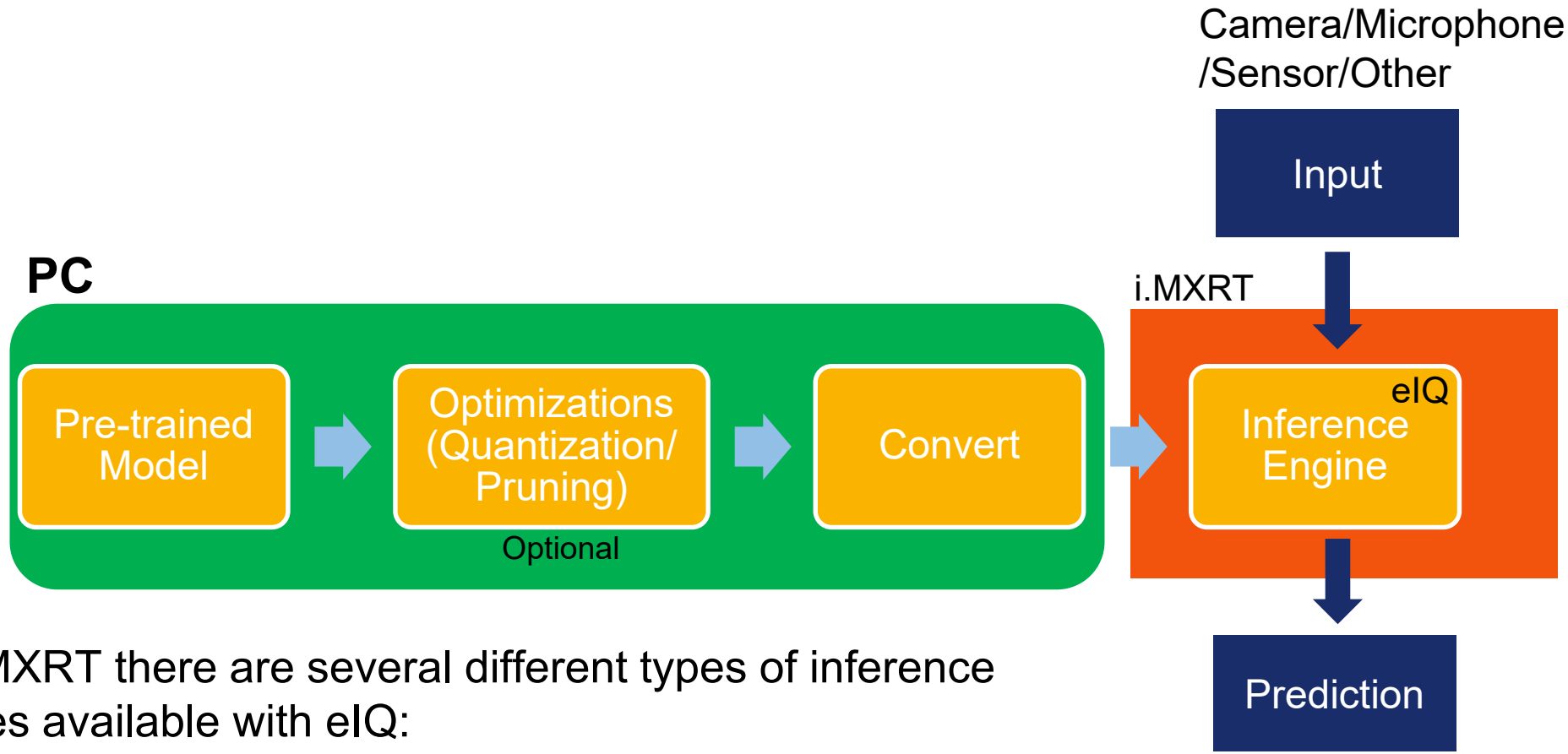
*Combination of parsed graph + CMSIS-NN

eIQ Component Diagram for MPUs/GPUs



EIQ ON I.MX-RT

eIQ for iMXRT



For i.MXRT there are several different types of inference engines available with 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 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 TENSORFLOW

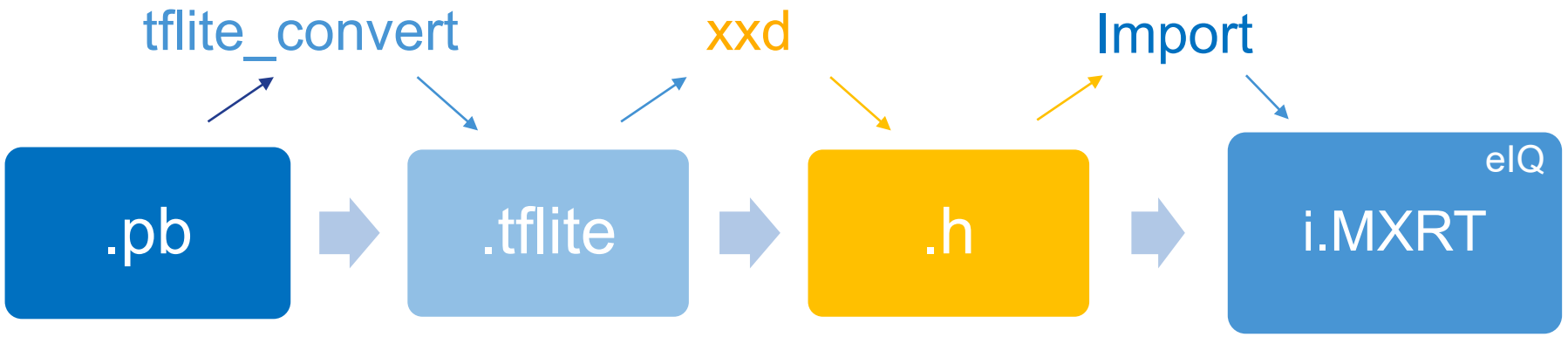
TensorFlow Lite Inference Engine

- Developed by Google
- Uses tflite_convert utility (provided by TensorFlow) to convert a TensorFlow model to a .tflite binary
- Load the .tflite binary into embedded system and use TensorFlow Lite inference engine running on i.MXRT to run model

- Only can be used for TensorFlow models.
- Tensorflow Lite supports a subset of Tensorflow operators.
  - Depending on model, conversion may not be possible or require custom implementation.
  - https://www.tensorflow.org/lite/guide/ops_compatibility

TensorFlow Lite Conversion Process

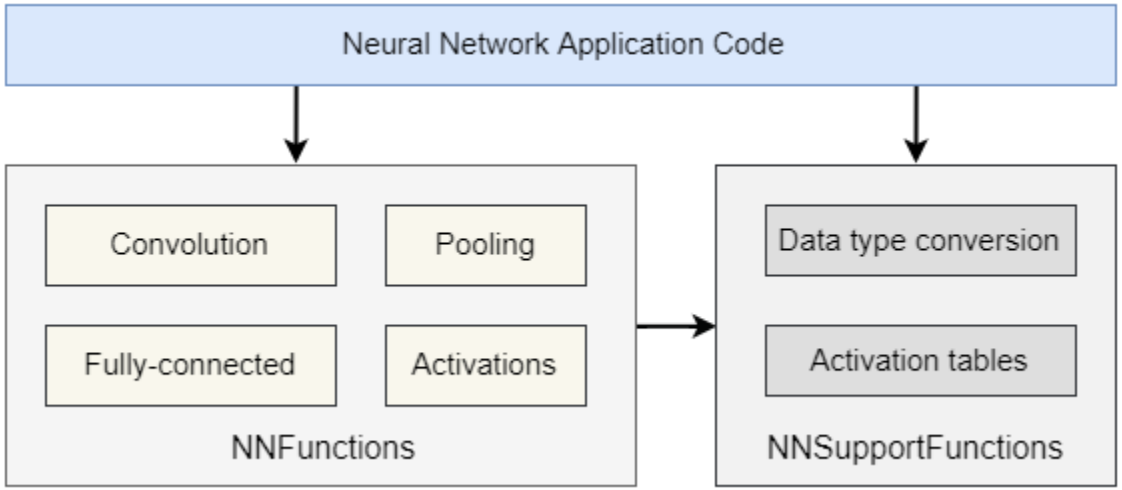
Transform a TensorFlow .pb model to a **.h header** to load model into embedded system with eIQ.



EIQ CMSIS-NN

CMSIS-NN Inference

- Developed by ARM
- API to implement common model layers such as convolution, fully-connected, pooling, activation, etc, efficiently at a low level
- Conversion scripts (provided by ARM) to convert Caffe models into CMSIS-NN API calls.
- CMSIS-NN could also be used to optimize the implementation of other inference engines
- Using “Release” high optimization compile settings significantly reduces inference time



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

Convolution

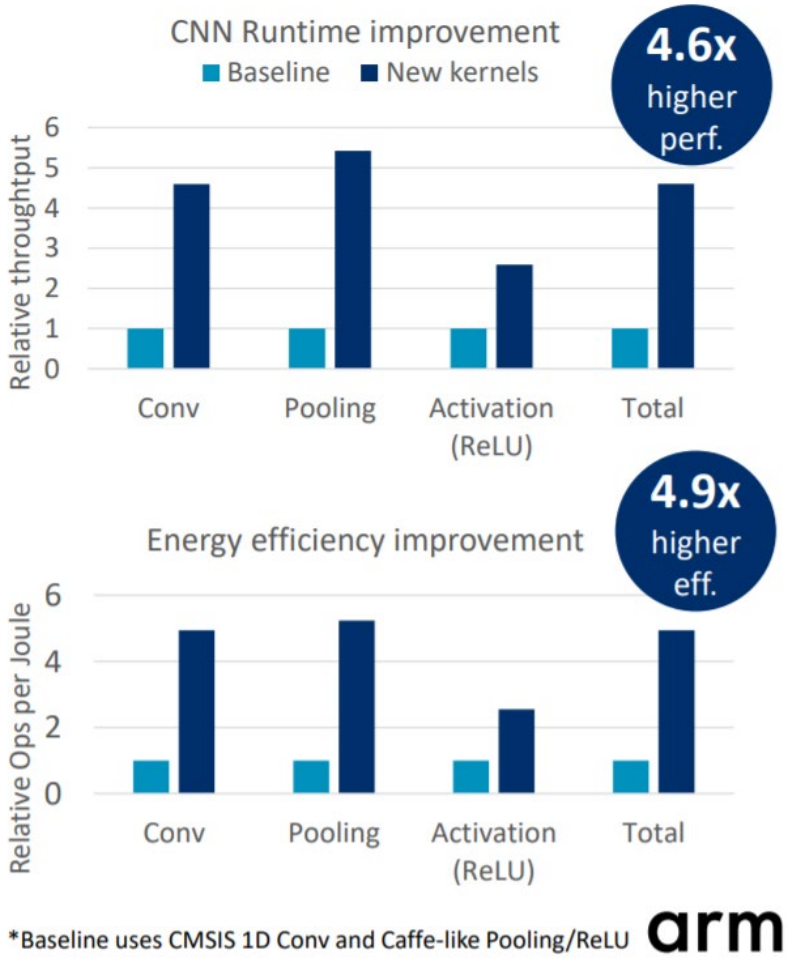
- Boost compute density with GEMM based implementation
- Reduce data movement overhead with depth-first data layout
- Interleave data movement and compute to minimize memory footprint

Pooling

- Improve performance by splitting pooling into x-y directions
- Improve memory access and footprint with in-situ updates

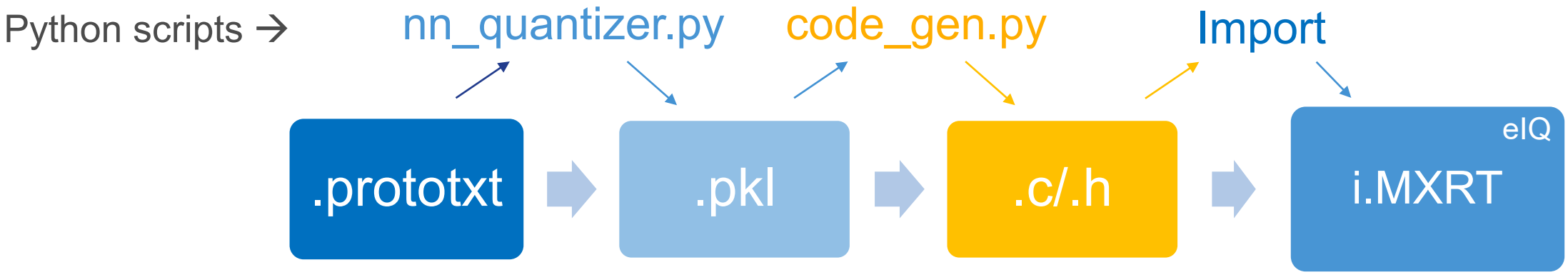
Activation

- ReLU: Improve parallelism by branch-free implementation
- Sigmoid/Tanh: fast table-lookup instead of exponent computation



CMSIS-NN Conversion Process (for Caffe Model)

Use Python to transform a **Caffe** model to **C files** to use in embedded system with eIQ.



EIQ GLOW

Glow (Coming Soon)

- Developed by Facebook
- Glow is a compiler that turns a model into machine executable code for the target device.
  - The model and the inference engine are built into the code that is generated.
  - Integrate the generated library into an SDK software project
- Cutting-edge inference technology

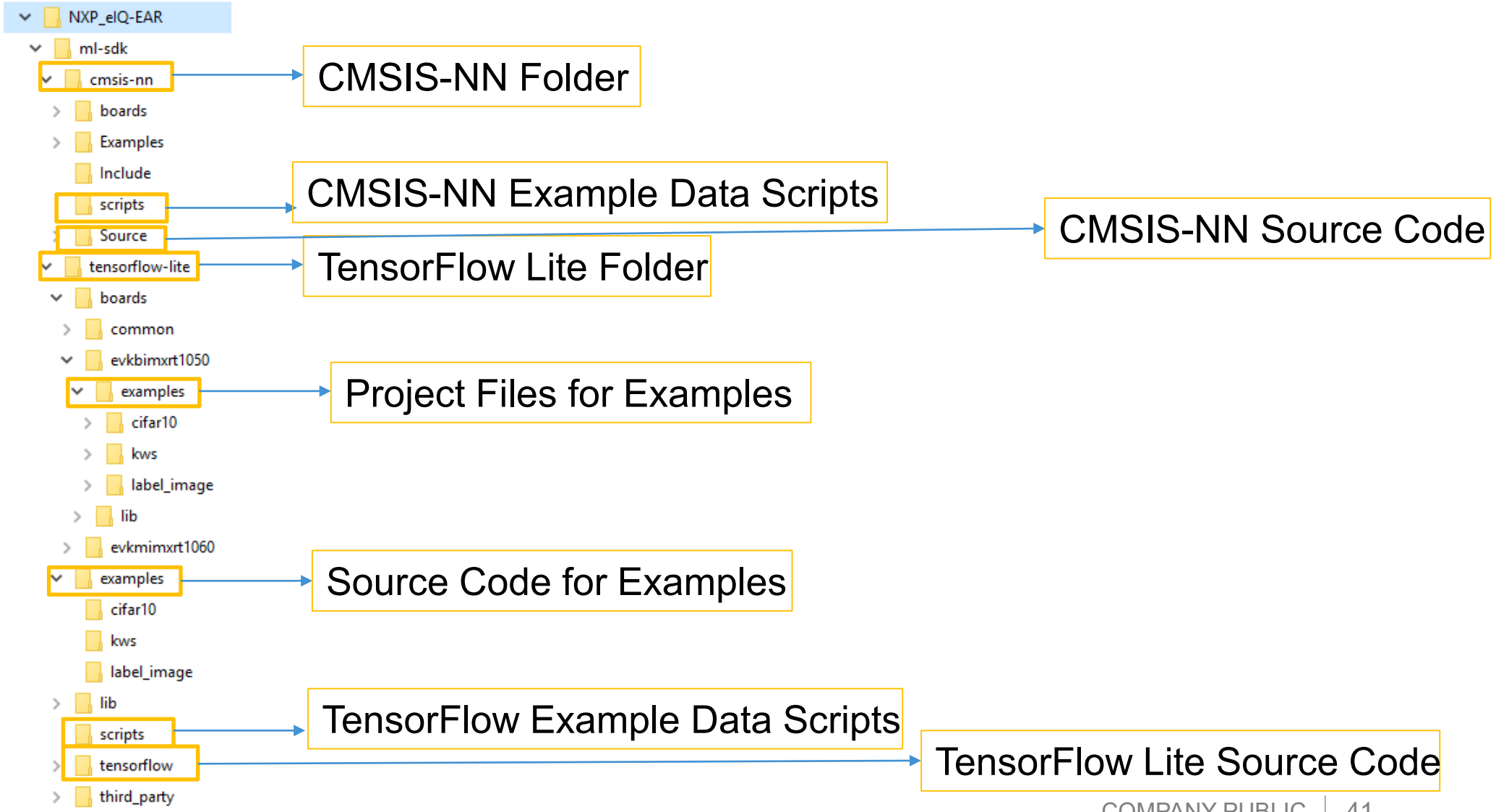
EIQ I.MX-RT EXAMPLES

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			

eIQ Folder Structure



Inference Times

- Benchmarking ongoing and optimizations still under development. Numbers subject to change.
- 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: 72ms

Memory Requirements

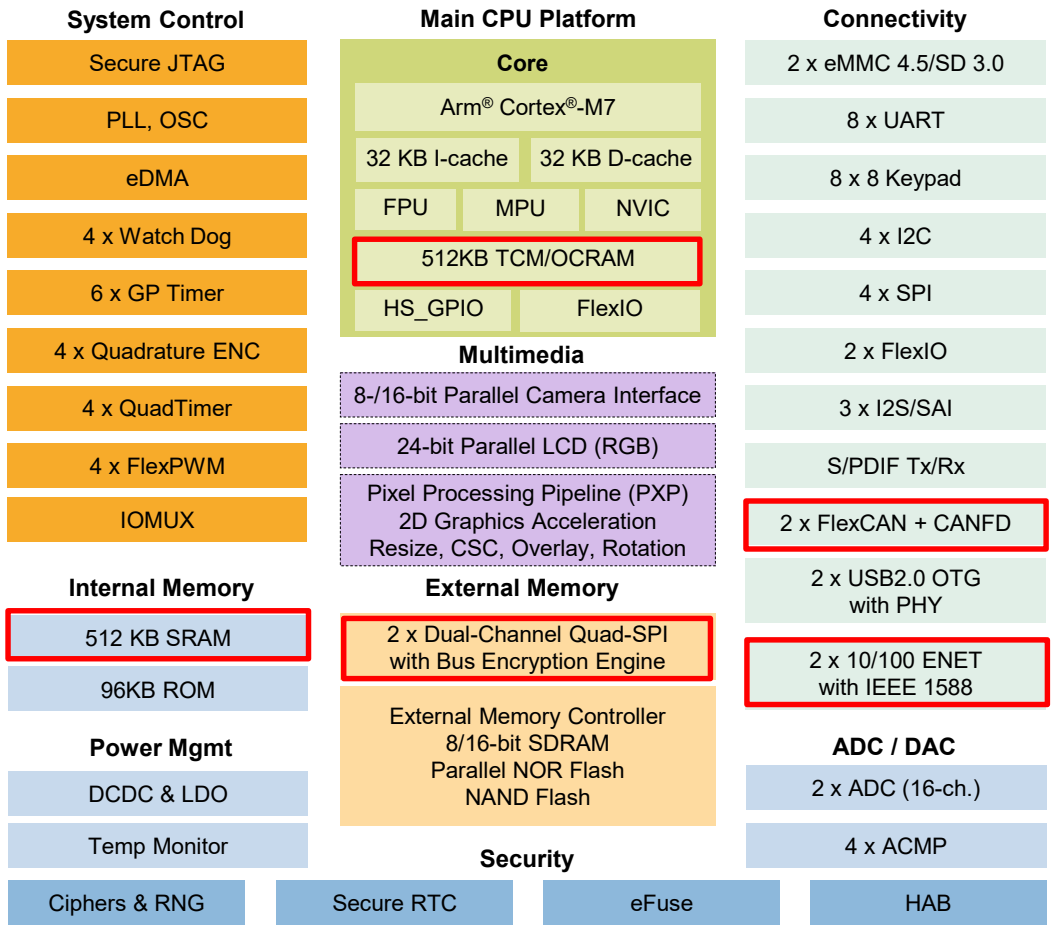
- Non-volatile memory (Flash/HyperFlash) stores the model, inference engine, and input data
- Volatile memory (SRAM/SDRAM) stores the intermediate products of the model layers
  - Amount required depends on a lot of factors like the amount, size, and type of the layers.

Benchmarking ongoing and optimizations still under development. Numbers subject to change:

- **CMSIS-NN with CIFAR-10:** 110KB Flash, 50KB RAM
- **TensorFlow Lite with CIFAR-10:** 600KB Flash (92KB for model, 450KB for inference engine), 320KB RAM
- **TensorFlow Lite with Label Image:** 1.5MB Flash (450KB for model, 450KB for inference engine, 450KB for input photo), 2.5MB RAM

NXP 32-bit Arm-based MCUs – High Performance

i.MX RT1060: Block Diagram



Available on certain product families

Specifications

- Package: MAPBGA196 | 10x10mm^2, 0.65mm pitch (130 GPIOs)
- Temp / Qual: -40 to 105°C (Tj) Industrial / 0 to 95°C (Tj) Consumer

High Performance Real Time system

- Cortex-M7 up to 600MHz , 50% faster than any other existing M7 products
- 20ns interrupt latency, a TRUE Real time processor
- 512KB SRAM + 512KB TCM/OCRAM

Rich Peripheral

- Motor Control: Flex PWM X 4, Quad Timer X 4, ENC X 4
- 2x USB, 2x SDIO, 2x CAN + 1x CANFD, 2x ENET with 1588, 8xUART, 4x SPI, 4X I2C
- 8/16-bit CSI interface and 8/16/24-bit LCD interface
- 2x Qual-SPI interface, with Bus Encryption Engine
- Audio interface: 3x SAI/ SPDIF RX & TX/ 1x ESAI

Security

- TRNG&PRNG(NIST SP 800-90 Certified)
- 128-AES cryptography
- Bus Encryption Engine: Protect QSPI Flash Content

Ease of Use

- MCUXpresso with SDK
- FreeRTOS
- Comprehensive ecosystem

Low BOM Cost

- Competitive Price
- Fully integrated PMIC with DC-DC
- Low cost package, 10x10 BGA with 0.65mm Pitch
- SDRAM interface

Transfer Learning and Inference Lab

- https://community.nxp.com/docs/DOC-343827

- Can take a pre-existing model and train it on new input
  - Allows much quicker training on an already known good model
  - Need to ensure model type is a good match for the type of data being retrained for
    - Some models better at image recognition. Others at speech.

- Lab will re-train a Mobilenet model built with TensorFlow to categorize 5 different flower types in images
  - This can be used then for any types of images you're interested in

WRAP-UP AND Q&A

Further Reading

- NXP eIQ
- TensorFlow Lite
- CMSIS-NN

Machine Learning Courses:

- Video series on Neural Network basics
- ARM Embedded Machine Learning for Dummies
- Google TensorFlow Lab
- Google Machine Learning Crash Course
- Google Image Classification Practica

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

NXP eIQ Resources

- eIQ for iMX and RT on MCUXpresso SDK Builder
  - https://mcuxpresso.nxp.com
- Available for i.MXRT1050 and i.MXRT1060 in SDK
  - Can also run on i.MXRT1064: https://community.nxp.com/docs/DOC-344225

QUESTIONS

SECURE CONNECTIONS
FOR A SMARTER WORLD

www.nxp.com