FLEXIBLE OPTIONS FOR INFERENCE IMPLEMENTATION AT THE EDGE

MARKUS LEVY
HEAD OF AI AND ML TECHNOLOGIES
NXP SEMICONDUCTORS

SECURE CONNECTIONS
FOR A SMARTER WORLD

Machine Learning Concepts

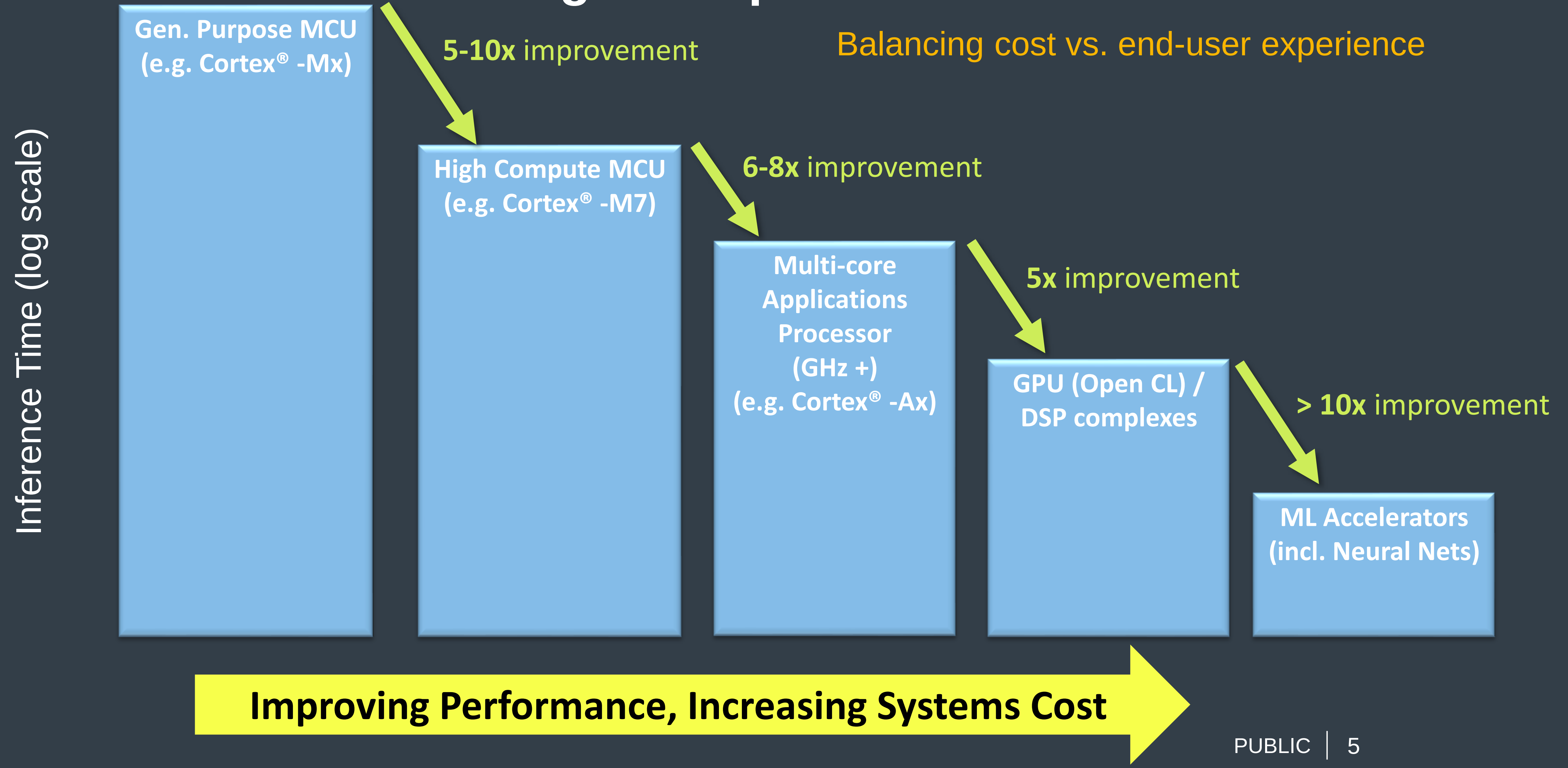
What is Machine Learning (ML)

- **ML** is a field of computer science (starting in 1960s) that gives computers the ability to learn **without being explicitly programmed**.
- It is not a single algorithm! It is not only Neural Nets.
- Biggest field of ML is supervised learning (learning with a teacher).
- In supervised learning an algorithm is provided with a set of examples – inputs and desired outputs.
- During training, an algorithm tries to minimize an error (on the output) by adjusting internal parameters.

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

Edge Compute Enabler – Scalable Inference



Processing unit comparison (Resnet-50)

	Size	Frequency	Inference/s	Cost efficiency
1x M7	1 (normalized)	600 MHz	1 (normalized)	1 (normalized)
4x A53	5.9	1.8 GHz	5.4	0.95
4x A55	8.3	1.8 GHz	33	4.0
Mid-range GPU	8.3	800 MHz	11	1.3
Gen 1 ML IP	3.3	1 GHz	350	106
Google TPU	550	750 MHz	~15000	27

Rule-of-Thumb ML Considerations

- Convolutional neural networks - object recognition, image and computer vision
- Recurrent neural networks - speech, handwriting recognition and time series
- Don’t consider training a deep neural net unless you have LOTS of training data.
- *Classical* ML model types can be trained with smaller data sets.

What can machine learning do

Regression (Calculation)

- Predict continuous values

Classification (Choice)

- Recognition, object detection

Anomaly detection (Judgement)

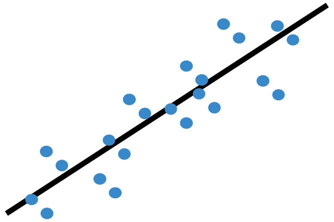
- Detect abnormal conditions

Clustering

- Discover patterns / partitions

Learn strategies

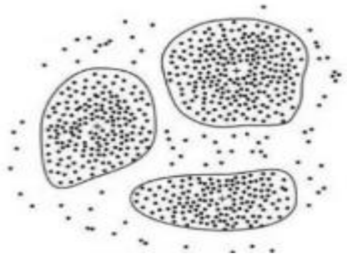
- Reinforcement Learning



X=a, y=?

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

Heart is going to malfunction? Y/N



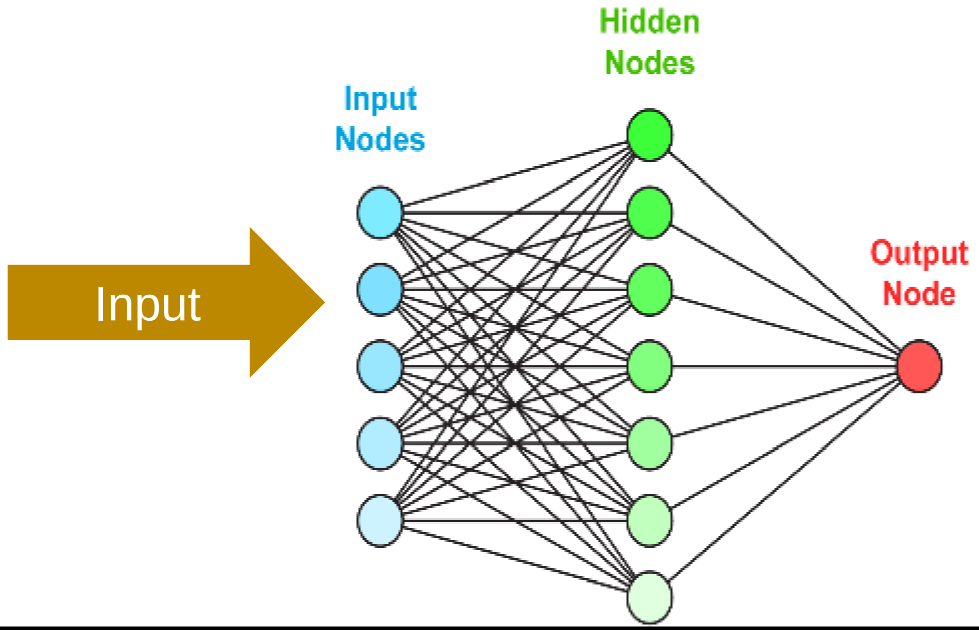
Find crowds
No need labels

How to play the game?

How To Speak ML and Sound Like an Expert:

The Neural Net inferred a label of ‘Dancing Banana Man’

With a confidence factor of 85%



85% A dancing banana man

10% Eyeballs on a peach slice

2% A moon rising over an island

1% A taco with cauliflower

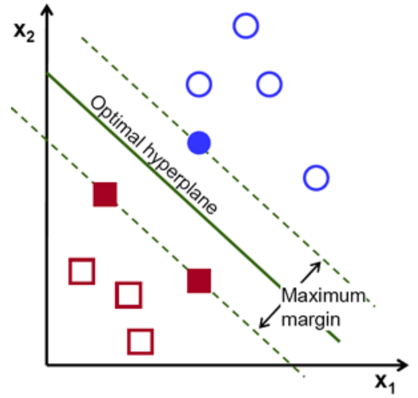
1% A banana

Neural Nets Infer/Predict a Label with a Confidence Factor
They Do Not Inherently ‘Decide’ What Something Is

What is Classical ML?

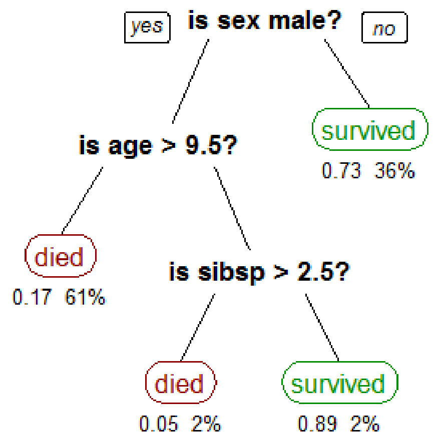
- Every ML algorithm except neural nets: SVM with linear and RBF kernels, Decision trees, Random forest, K-Nearest neighbors, Boosting algorithm (ada-boost), Logistic regression, k-means
- Usually much smaller number of parameters and don’t need big training datasets
- Usually faster (both training and inference) compared to NNs
- Might be used in combination with NNs
- Most of the algorithms require careful feature selection

SVM

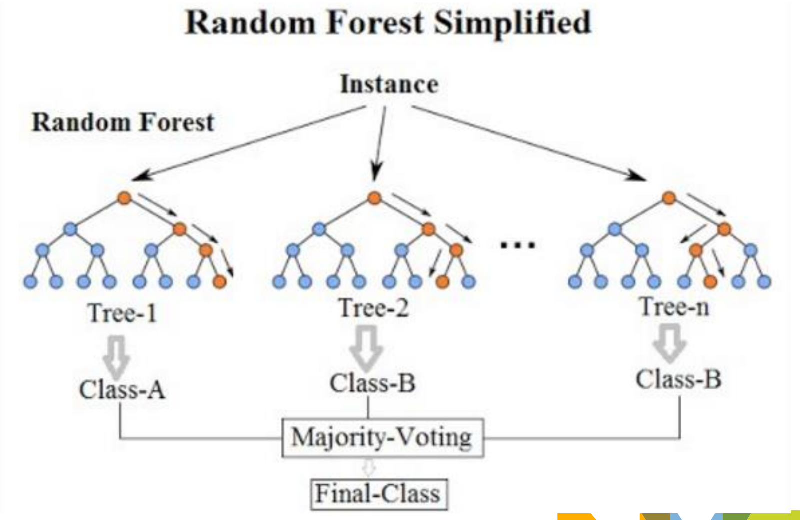


https://docs.opencv.org/2.4/_images/optimal-hyperplane.png

Decision Tree



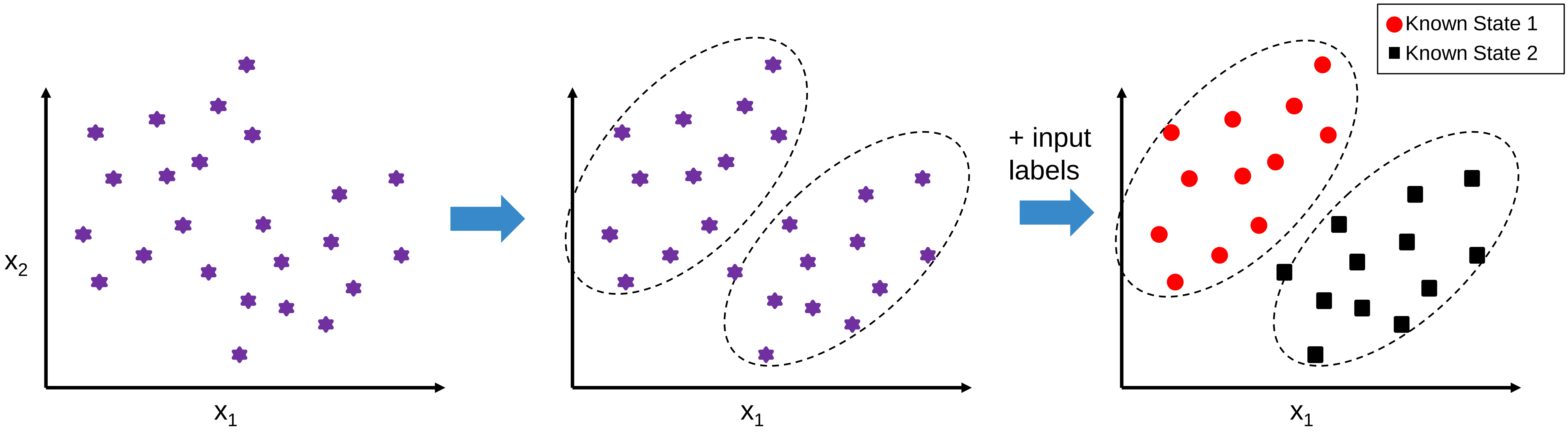
https://en.wikipedia.org/wiki/Decision_tree_learning



https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

Supervised vs Unsupervised

- Supervised – Given X_i and Y_i compute f where $Y_i = f(X_i)$
- Unsupervised – Given only X_i , find the patterns

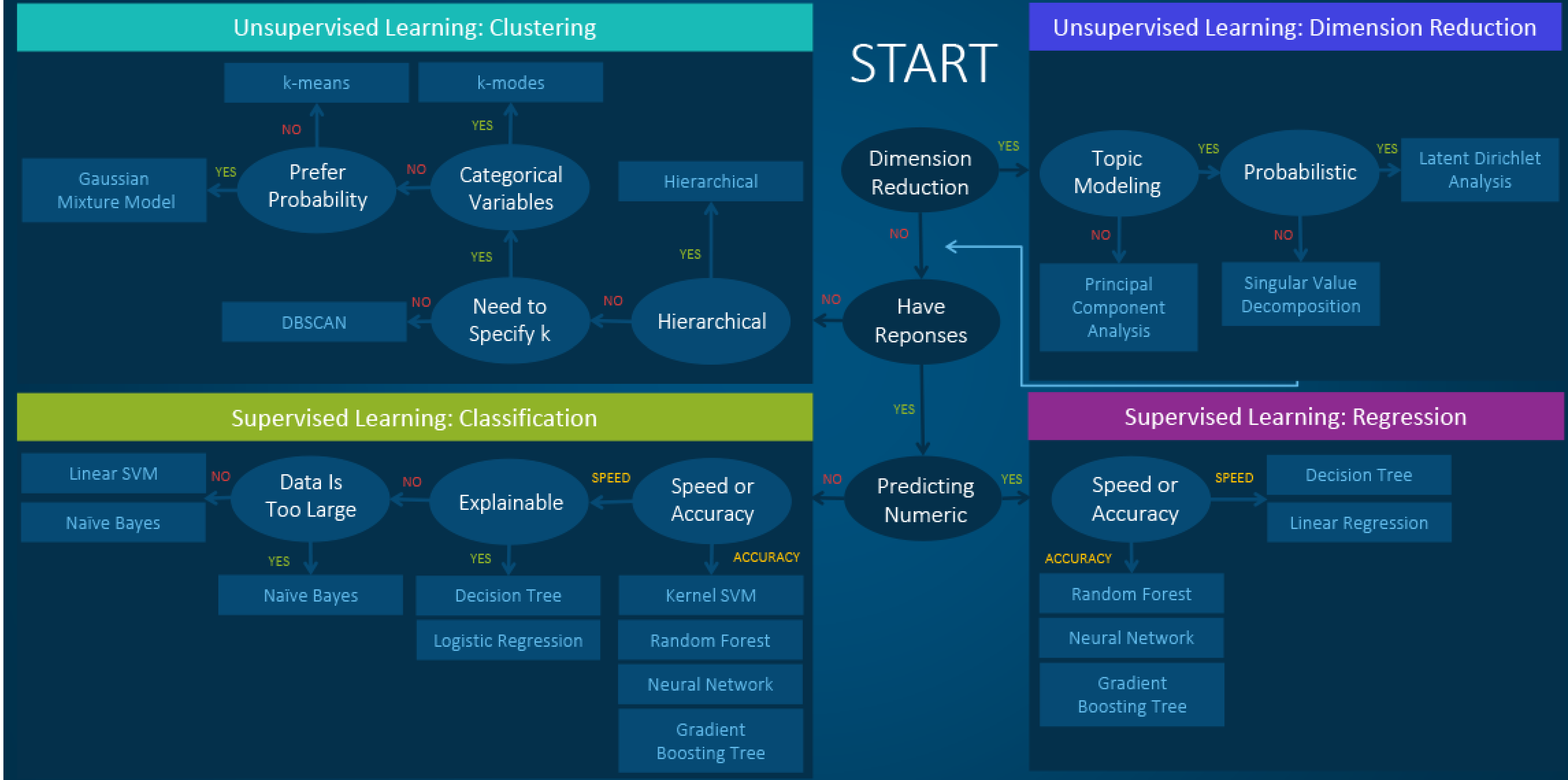


Raw Data

Unsupervised → you can cluster, but not identify cluster label

Supervised → you can fully classify

Machine Learning Algorithms Cheat Sheet



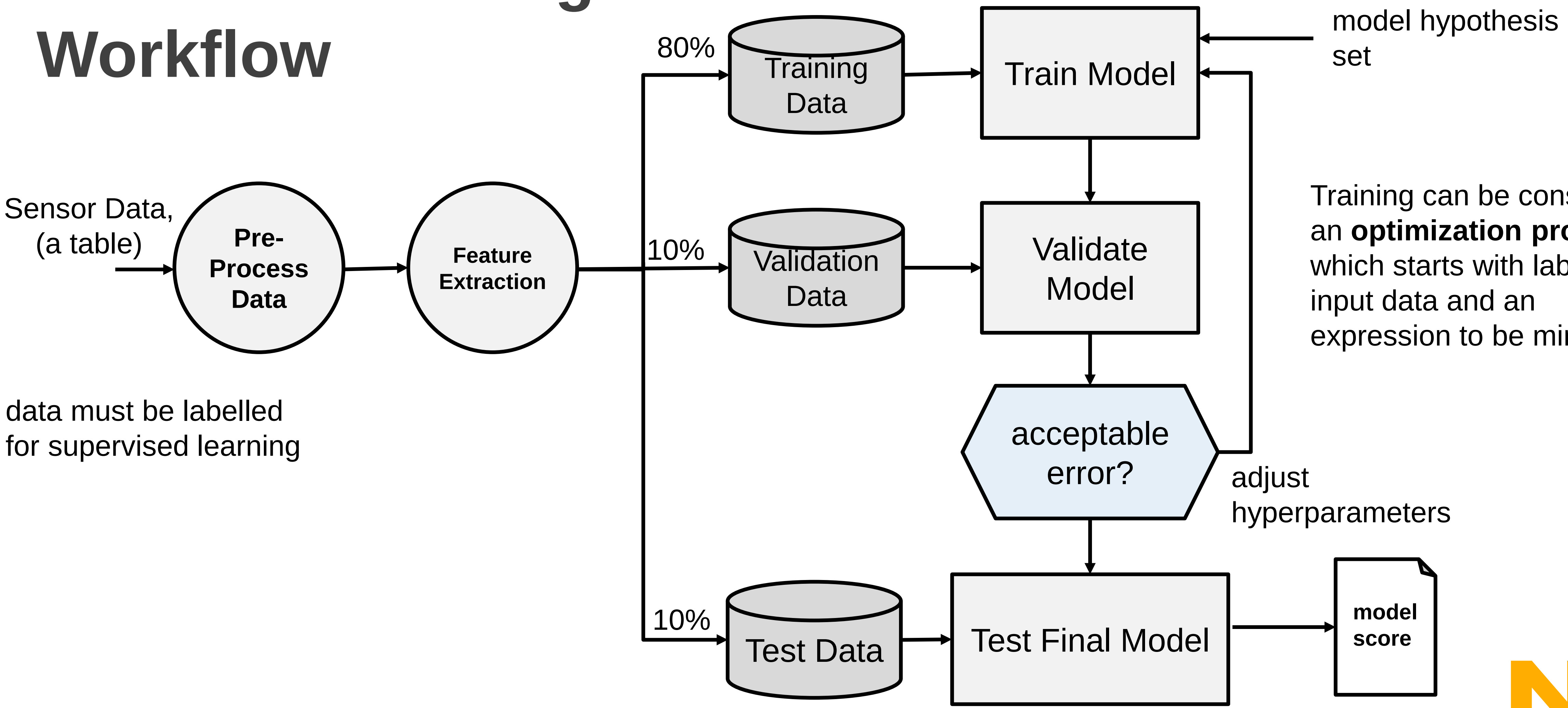
https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/

Traditional Machine Learning Workflow

Training data is pseudo-randomly chosen

Validation used to tune hyperparameters

Test to evaluate and predict generalization for new data



data must be labelled for supervised learning

Training can be considered an **optimization problem** which starts with labelled input data and an expression to be minimized

Trained Model Optimizations for Mapping to HW Capabilities

Quantize parameters - 32-bit floating point to 8-bit fixed-point -> 4x memory reduction

- Weights can be shared to improve compression

Operation fusing

- Fuse or chain operations to avoid roundtrips to accelerators
- Next gen NN supports operations for: convolution, activation, pooling and normalization

Pruning (sparsity)

- Remove weights and neurons w/small impact on accuracy, reducing memory size 4-10x
- Requires additional training

Next gen IP supports decompression scheme to further reduce weights memory footprint

Move from the Cloud to the Edge

Cloud Access With Amazon & Google ML Services

1. AWS SageMaker

Build, train & deploy ML models

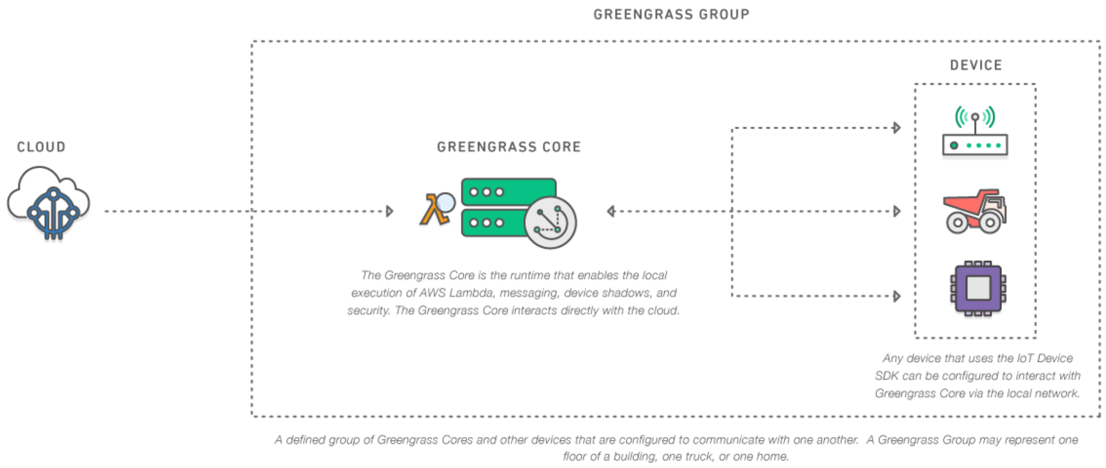
Amazon SageMaker

2. AWS HyperParameter Optimization

Optional – to achieve better accuracy

3. AWS Greengrass ML – IoT service

Train on Cloud, Deploy on Edge



1. Google ML Engine

- Training & predictions services
- TensorFlow support

2. Google AutoML

- Designed for beginners/users which want to obtain fast a model
- Based on dataset is able to build a NN, train it and obtain a model
- 2 flavors
  - Based model (for free)
  - Advanced model (550$)

CLOUD AUTOML VISION

Google Cloud Interoperability

Cloud cookbook details interoperability between Cloud and ML SDK w/OCV

- Train using Google Cloud
- Deployed on i.MX 8 using OpenCV DNN

Instructions to teach user how to

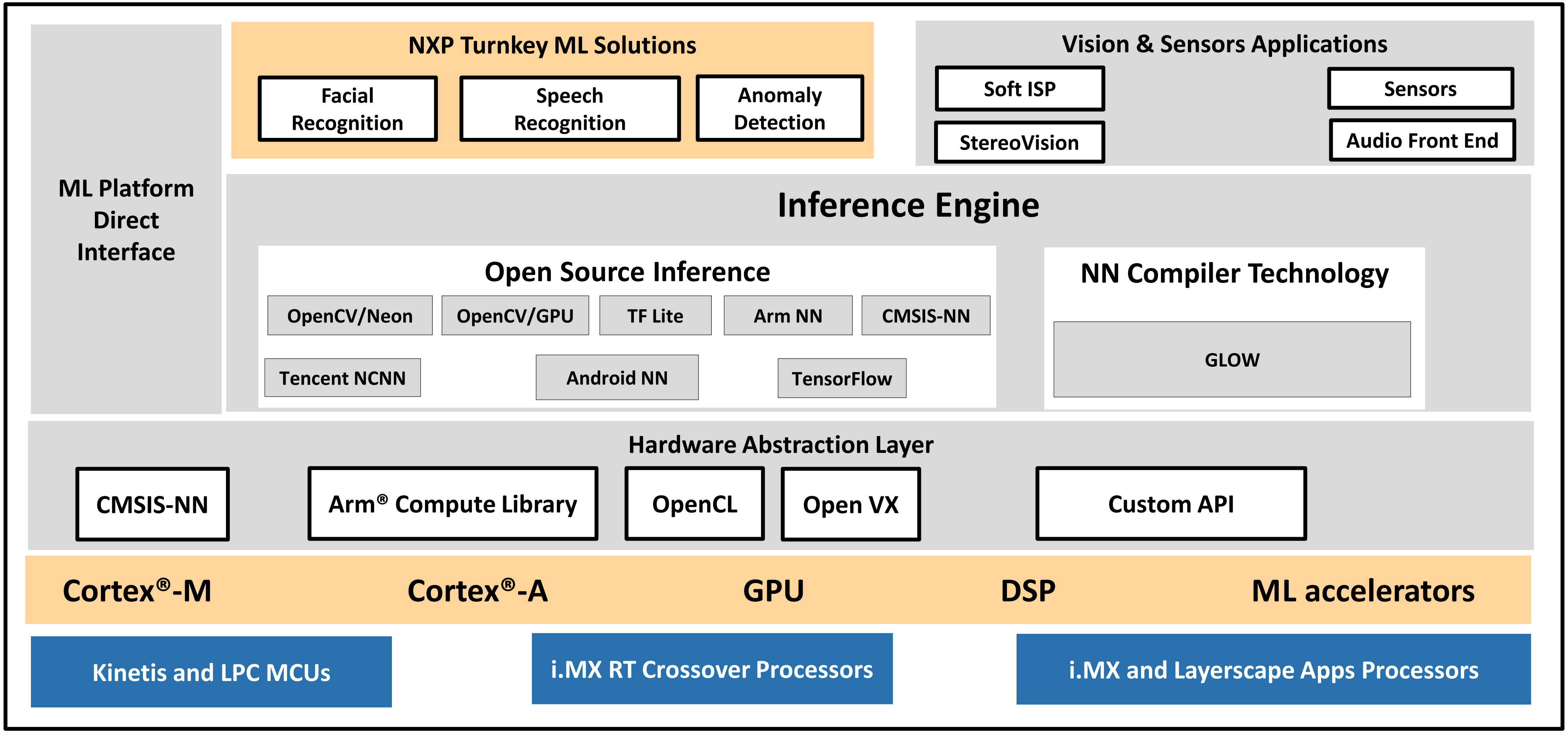
- train a neural network (written in TensorFlow) on Google Cloud
- use the ML service
- store the model on Google Cloud storage
- download it locally
- use the Cloud model to perform inference locally

Google Cloud Platform

Edge Device

Machine Learning Deployment Overview

NXP eIQ Machine Learning Software Development Environment



Machine Learning Deployment

(The Easy Way)

Open Source Computer Vision Library: http://opencv.org

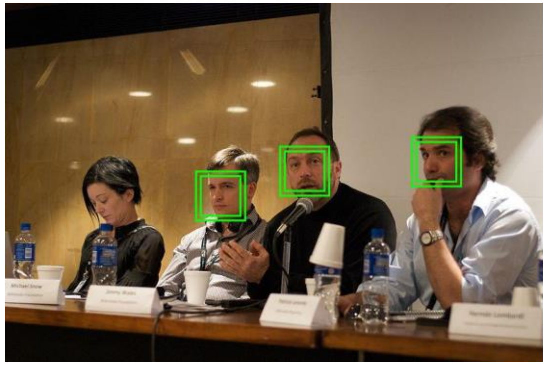
- Open-source BSD-licensed library

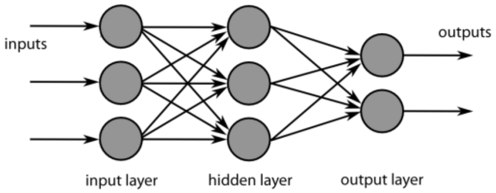
- Includes several hundreds of computer vision algorithms
  - Image processing, image encoding/decoding
  - Video encoding/decoding
  - Video analysis
  - Object detection
  - Deep neural networks
  - Machine learning

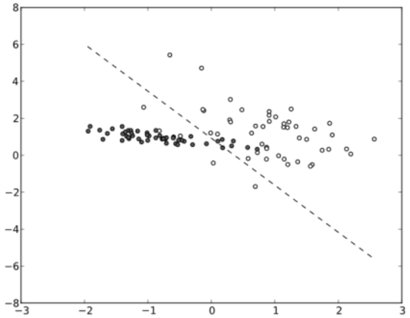
- Supports ARM NEON and OpenCL for acceleration

OpenCV introduction

- Can be used in combination with deep neural networks
  - Example: facial recognition







Face detection using OpenCV object detection

Feature extraction using deep neural network

Face classification using OpenCV machine learning

New AppNote

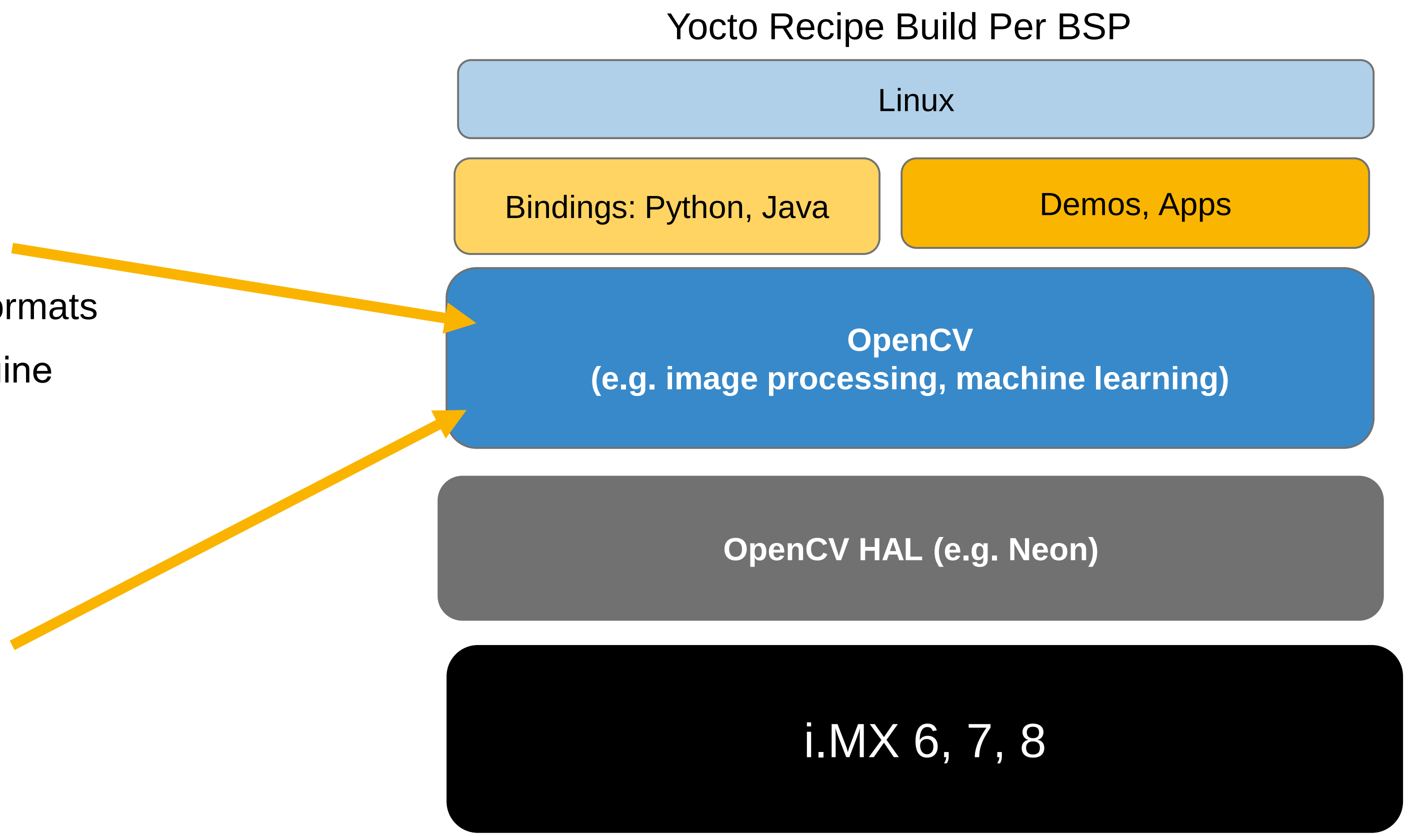
NXP Semiconductors
Application Note

Document Number: AN12224
Rev. 0, 07/2018

ML SDK with OpenCV

ML SDK with OpenCV 1.0

- OpenCV DNN Module
  - Inputs Caffe/TensorFlow formats
  - Provides NN inference engine
  - Optimized for Neon
- OpenCV ML Module
  - Classical ML algorithms
  - Optimized for Neon

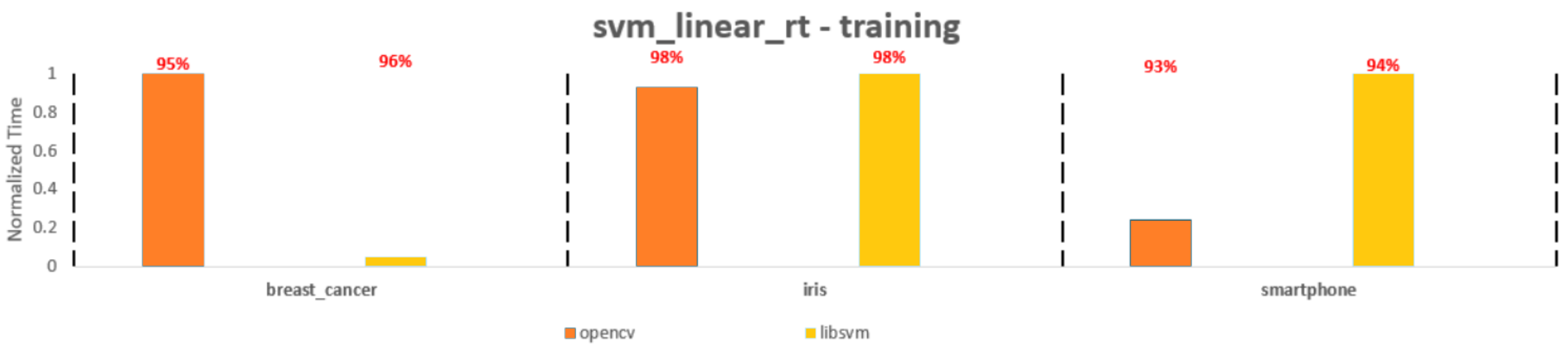


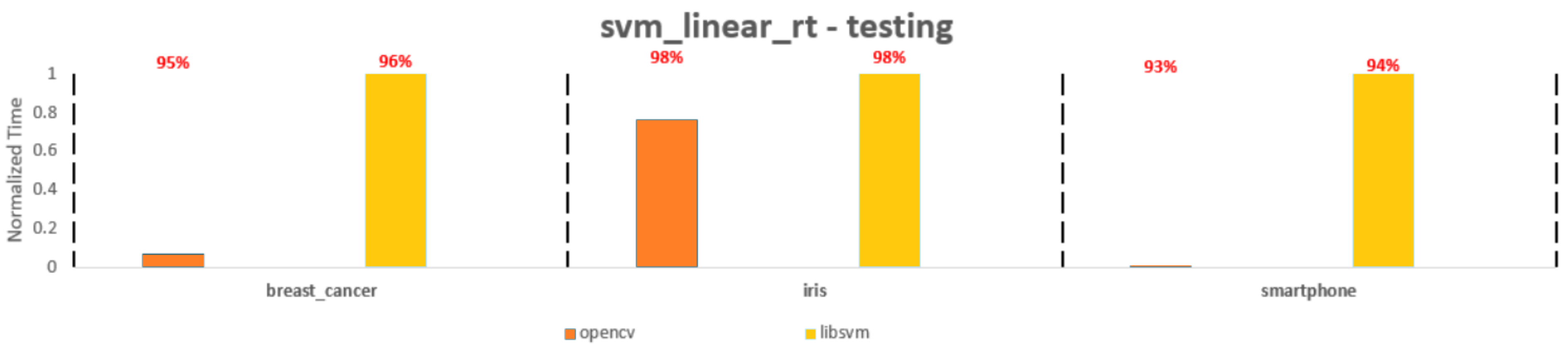
Documentation provides scripts & detailed description to modify **Caffe** and **TensorFlow** models to run inference using OpenCV

Why OpenCV for CML

	OpenCV	Dlib	mlpack	shark	shogun	H2O	Libsvm	liblinear	svm^perf	ThunderSVM
SVM (linear)	x	x	-	x	x		x	x	x	x
SVM (RBF)	x	x	-	x	x		x		-	x
Decision Trees	x	-	x	x	x					
Gradient Boosting	x	-			x	x				
EM (GMM)	x	-	x		x					
Logistic Regression	x	-	x		x	x		x		
AdaBoost (ml::Boost)	x	-	x							
Random Forests	x	x	x	x	x	x				
KNN	x	x	x	x	x					
k-means	x	x	x	x	x	x				
NEON support	x	x	-	x	-		-	-	-	x

Training and Inference Performance on M7 (e.g. i.MX RT)



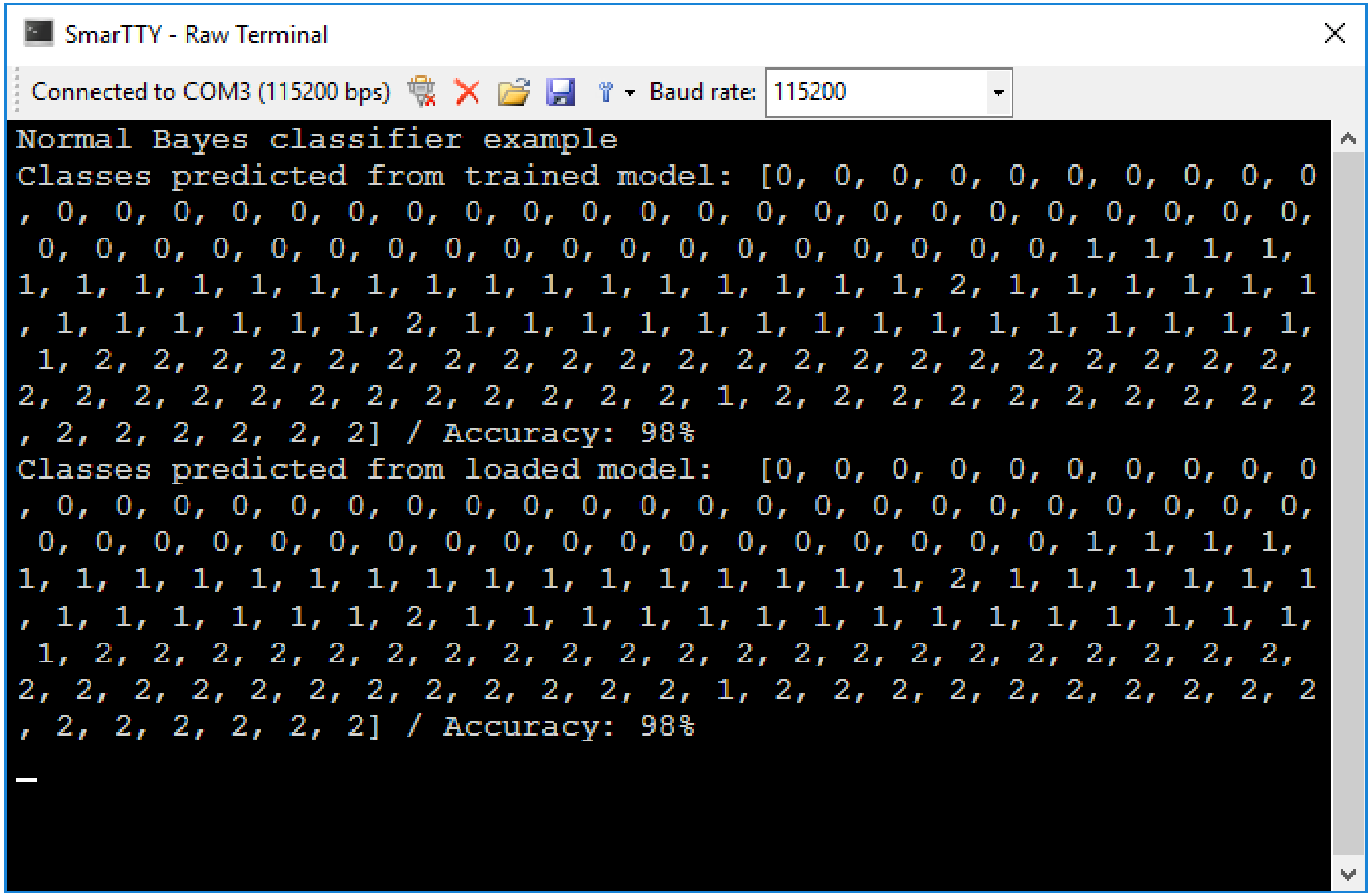


Notes:
1. For training, OCV almost 2 orders of magnitude slower than libsvm due to some problem with class separability; could be solved by using RBF kernel, but we haven’t done measurements with that (refer to benchmarking presentation).
2. OCV is faster on testing in all cases, and even 2 orders of magnitude faster on smartphone data

Training Can Be Done in a Few Function Calls

```
#include <opencv2/core/core.hpp>
#include <opencv2/ml/ml.hpp>
...
using namespace cv;
using namespace cv::ml;
...
    Mat samples = Mat_<float>(150, 4, samplesData);
    Mat labels = Mat_<int>(150, 1, labelsData);
    Mat_<int> responses;

    /* Prepare training data and labels */
    Ptr<TrainData> trainData = TrainData::create(samples, ROW_SAMPLE, labels);

    /* Create a model */
    Ptr<NormalBayesClassifier> trainedModel = NormalBayesClassifier::create();

    /* Train the model */
    trainedModel->train(trainData);

    /* Predict values */
    trainedModel->predict(samples, responses);
    cout << "Classes predicted from trained model: " << responses.t();
    cout << " / Accuracy: " << (countNonZero(responses == labels) / (float)labels.rows) * 100.0 << "%" << endl;
```

Super Boring Example Output



It's just a "Hello world". It demonstrates that it works – model training, loading, prediction.

Anomaly Detection as a Subset of Machine Learning

It’s All About the Data

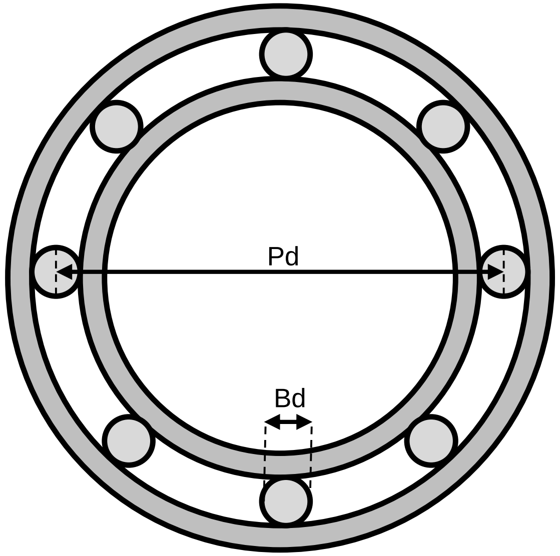
- Multi-class supervised learning requires representative data for all classes

- In machine condition monitoring applications, this can be impractical to get
  - Hard to run machinery to failure, certainly not a statistically significant number of times

- Enter “Anomaly Detection”, essentially a one-class learning problem
  - Only needs “nominal” data for training!!!

- The Goal:
  - Given a sample point X, compute the likelihood that X is a member of population all_X’s.
  - Compare that to a specified threshold to determine if you have a nominal sample or not

Bearing Faults Have Specific Frequency Signatures



For ball defects:

BSF $= \frac{1}{2} (P_d/B_d) \times S \times [1 - (B_d/P_d \times \cos\theta)^2]$

For outer trace defects:

BPFO $= \frac{1}{2} N_b \times S \times [1 - (B_d/P_d \times \cos\theta)]$

For inner trace defects:

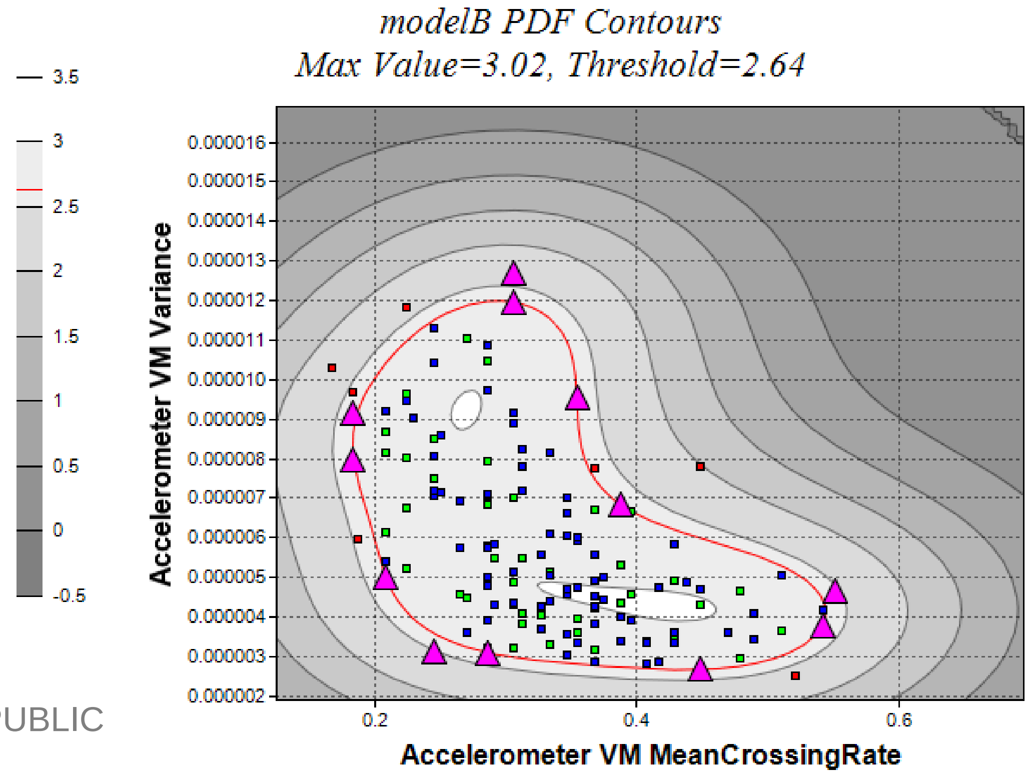
BPFI $= \frac{1}{2} N_b \times S \times [1 + (B_d/P_d \times \cos\theta)]$

P_d = pitch diameter
B_d = ball diameter
N_b = number of balls
S = speed (revolutions/sec)
θ = contact angle
BSF = Ball Spin Frequency
BPFO = Ball Pass Frequency of Outer Trace
BPFI = Ball Pass Frequency of Inner Trace

Defect signals may be swamped by other noise in the system, in which case additional filtering may be needed to extract the signature.

One Class Support Vector Machines

- Used for anomaly detection
- The algorithm tells us if a sample is part of a known population or not
- Computing a probability by comparing with a threshold value
  - Each contour line corresponds to a different threshold



We are using a Gaussian Kernel

$$f_{svm}(x) = \sum_{i=1}^{n} \alpha_i e^{-\frac{\|x - sv_i\|^2}{2\sigma^2}}$$

where:

x is a d-dimensional feature vector
$\{sv_i\}_{i=1}^{n}$ are support vectors (SVs)
$\{\alpha_i\}_{i=1}^{n}$ are coefficients for SVs
σ is known as the “kernel size”

a sample is considered True (1) if pdf $f_{svm}(x)$ > threshold or False (-1) otherwise

$$sgn[f_{svm}(x) - thr]$$

Product Life Cycle Intelligence – Monitoring/Tracking Use Cases

Factory – Key/ certificates provisioning, datalogging, movement monitoring, manufacturing and self tests (BLE/Wi-Fi/NFC)

Transport to store or warehouse

Review transport data, display/demo, or storage

Ship to consumer

← Continuous sensor logging (Battery powered) →

Installer reviews transport data, runs self-tests, securely onboards device w/cloud, initiates run-time data collection

Secure run-time data upload. OEM uses data to further tune ML models, add capabilities.

Secure periodic ML model &/or general SW capability updates

Continuous secure monitoring & cloud upload

Data initiated preventative maintenance request

Preventive maintenance, self-test w/ data history sent to cloud

Longer product life due to preventive maintenance and periodic capability upgrades. E.g. "Clean Clothes as a Service".

End-of-life decommissioning and credentials recovery

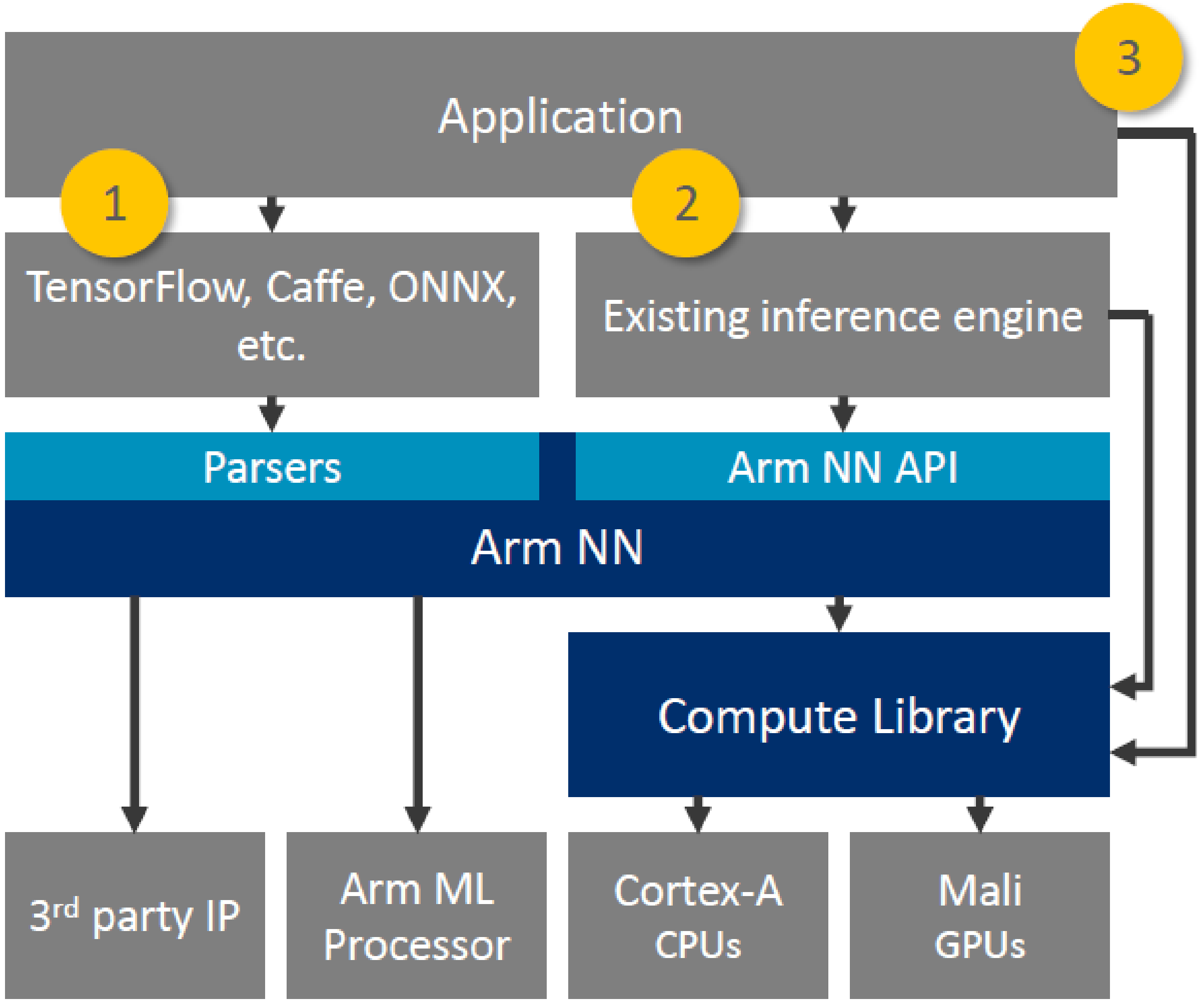
Other Open Source Options

Deployment of Arm NN

1. Connect to Arm NN through high level frameworks
   - Using framework parsers provided by Arm NN

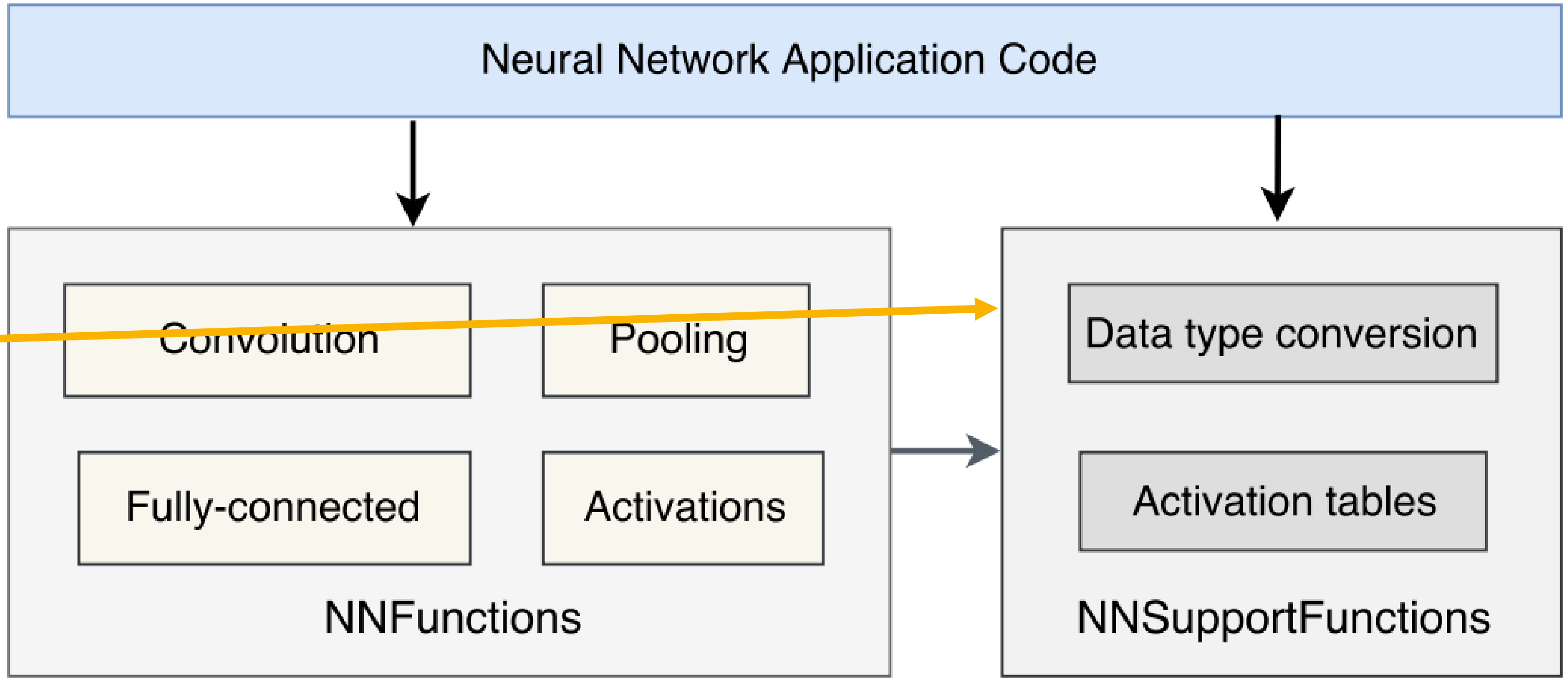
2. Connect to existing inference engine
   - With inference engine calling Arm NN API
   - Or inference engine calling ACL directly

3. Connect to ACL directly



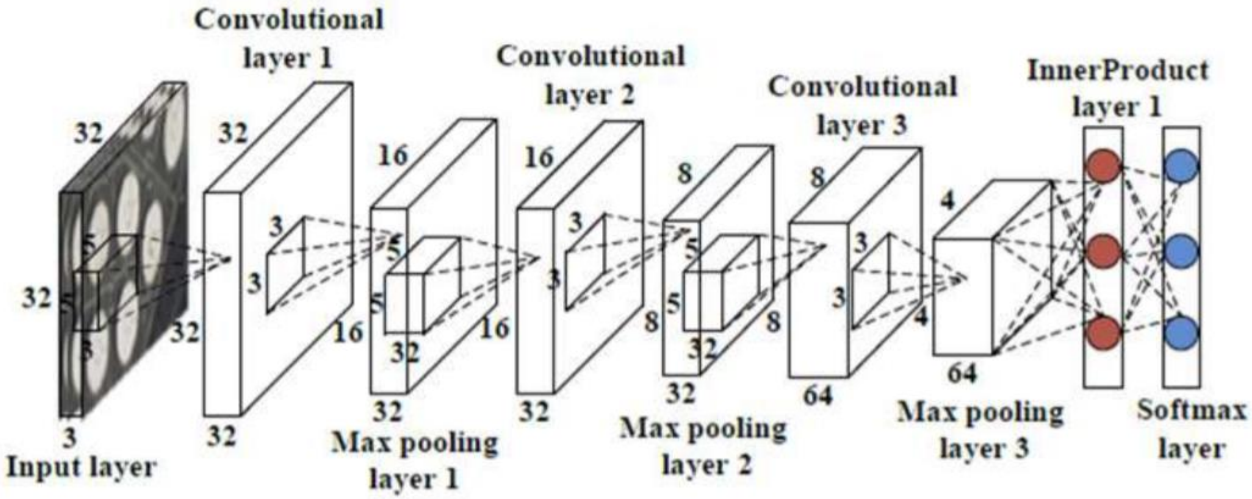
CMSIS-NN – Efficient NN Kernels for Cortex-M CPUs

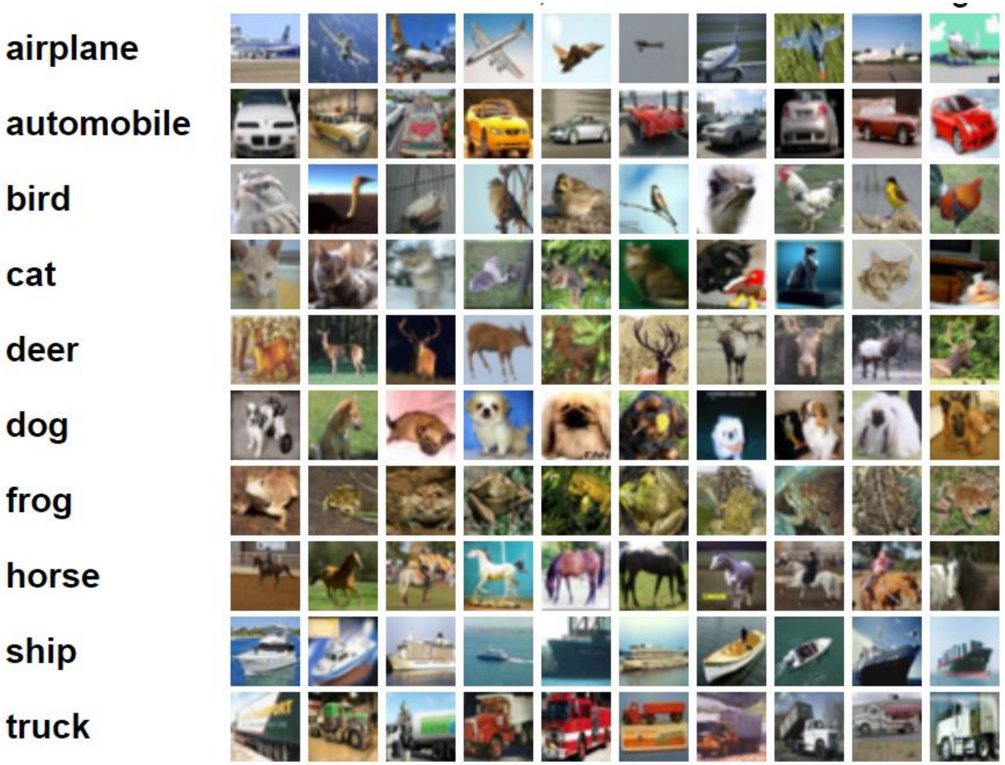
- CNN library for Cortex-M by ARM, new in CMSIS 5.3
- Fixed-point inference
- High level API
- Low level API
- Make use of some CMSIS math & DSP lib APIs.
- **4.6x performance & 4.9x energy efficiency than baseline CMSIS-DSP**



CIFAR-10 model

- CIFAR-10 classification – classify images into 10 different object classes
- 3 convolution layer, 3 pooling layer and 1 fully connected layer (~80% accuracy)
- https://www.cs.toronto.edu/~kriz/cifar.html





Inference engine based on CMSIS-NN

dataset input for training

DL frameworks (Google TensorFlow&Keras, Caffe/Caffe2, Facebook PyTorch, Amazon MxNet, ONNX, etc)

trainer, quantizer, converter

PC tools to generate model

MCU firmware: inference engine

Data input: **video**,audio/time-series, structured data

Data feeder and preprocessor

Model runner (load, parse, and inference)

Loadable NN models: Can be stored in SD card, or be flattened to C arrays as source file.

Map model operations to CMSIS-NN APIs

CMSIS-NN adapter

NN SDK responsibilities

PC side : PC tool set

Firmware side: inference engine

CMSIS-NN

Silicon

Cortex-M parts

Benchmark Results using CMSIS-NN with CIFAR-10 Model

- Cortex-M4F(LPC54114) 212mS
- Cortex-M33(LPC55s69) 179mS

IDE

- IAR 8. 30.1
  - High / Speed / No size constrains

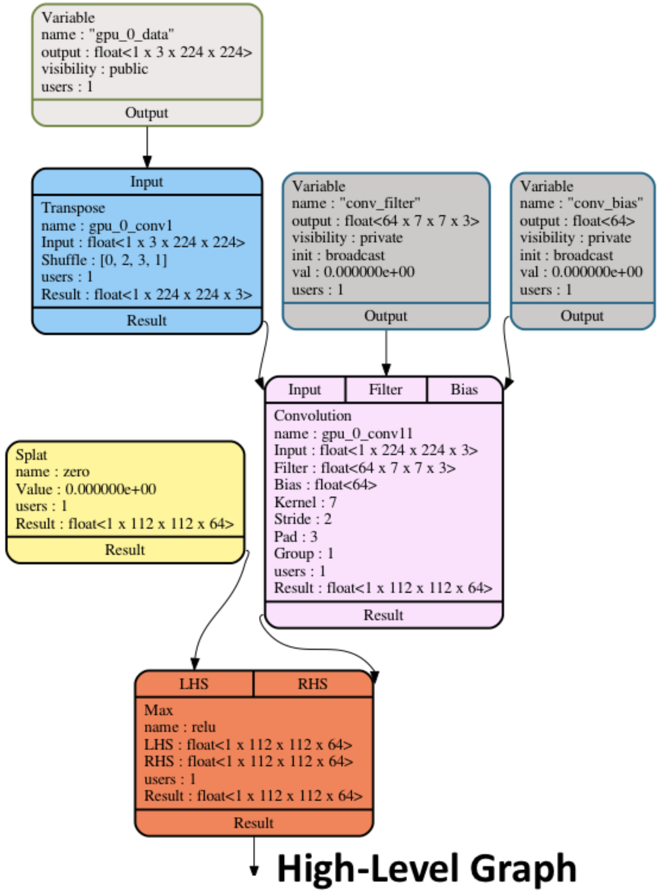
Video

Advanced Techniques

GLOW – Graph Lowering Compiler

- Facebook open-sourced Glow in March 2018
- Machine learning compiler to accelerate the performance of deep learning frameworks
- More rapidly design and optimize new silicon products for AI and ML by leveraging community-driven compiler software.
- Glow accepts computation graphs from a variety of machine learning frameworks and works with a range of accelerators.

GLOW – From Graph to Machine Code



High-Level Graph

```
declare {
 %input = weight float<8 x 28 x 28 x 1>, broadcast, 0.0
 %filter = weight float<16 x 5 x 5 x 1>, xavier, 25.0
 %filter0 = weight float<16>, broadcast, 0.100
 %weights = weight float<10 x 144>, xavier, 144.0
 %bias = weight float<10>, broadcast, 0.100
 %selected = weight index<8 x 1>
 ...
 %result = weight float<8 x 10>
}

program {
 %allo = alloc float<8 x 28 x 28 x 16>
 %conv = convolution [5 1 2 16] @out %allo, @in %input,
     @in %filter3, @in %bias0
 %allo0 = alloc float<8 x 28 x 28 x 16>
 %relu = max0 @out %allo0, @in %allo
 %allo1 = alloc index<8 x 9 x 9 x 16 x 2>
 %allo2 = alloc float<8 x 9 x 9 x 16>
 %pool = pool max [3 3 0] @out %allo2, @in %allo0,
     @inout %allo1
 ...
 %deal6 = dealloc @out %allo6
 %deal7 = dealloc @out %allo7
 %deal8 = dealloc @out %allo8
 %deal9 = dealloc @out %allo9
}
```

Low-Level IR

```
LBB14_1:
  vmovaps 3211264(%rcx,%rax,4), %ymm1
  vmovaps 3211296(%rcx,%rax,4), %ymm2
  vmovaps 3211328(%rcx,%rax,4), %ymm3
  vaddps 6422528(%rcx,%rax,4), %ymm1, %ymm1
  vaddps 6422560(%rcx,%rax,4), %ymm2, %ymm2
  vmovaps 3211360(%rcx,%rax,4), %ymm4
  vaddps 6422592(%rcx,%rax,4), %ymm3, %ymm3
  vaddps 6422624(%rcx,%rax,4), %ymm4, %ymm4
  vmaxps %ymm0, %ymm1, %ymm1
  vmaxps %ymm0, %ymm2, %ymm2
  vmaxps %ymm0, %ymm3, %ymm3
  vmovaps %ymm1, 6422528(%rcx,%rax,4)
  vmovaps %ymm2, 6422560(%rcx,%rax,4)
  vmaxps %ymm0, %ymm4, %ymm1
  vmovaps %ymm3, 6422592(%rcx,%rax,4)
  vmovaps %ymm1, 6422624(%rcx,%rax,4)
  addq $32, %rax
```

Machine Code

IoT Solutions

Putting it All Together

ML-Related Functions That Can Be Done on i.MX 8M Mini

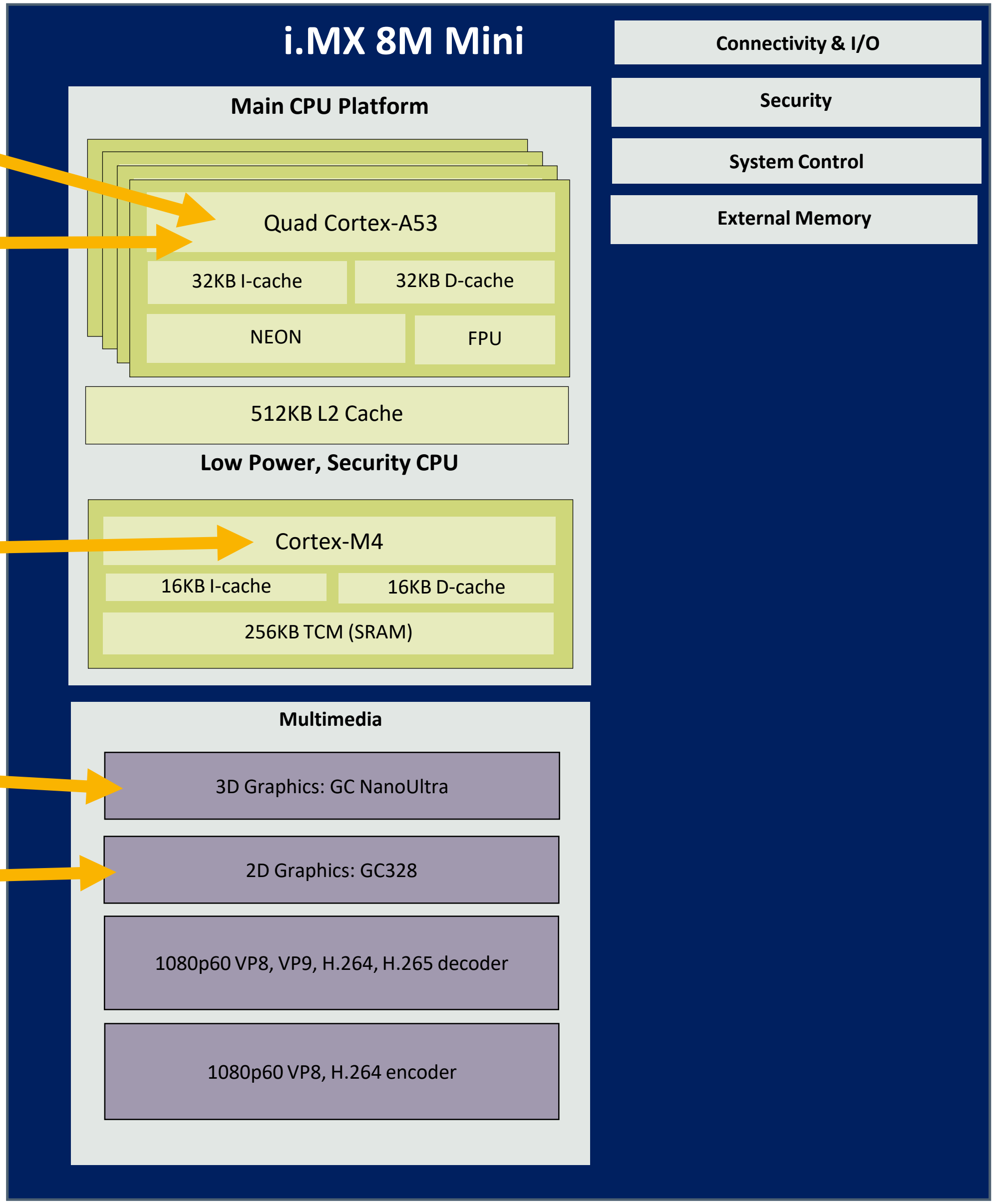
- OpenCV support accelerated on NEON
  - TensorFlow and Caffe
  - Classical machine learning algorithms

- Other open source options
  -Arm NN w/NEON acceleration using Arm Compute Library (ACL)
  -Android NN
  -TensorFlow, TF Lite (direct deployment)
  -EdgeScale deployment thru docker images
  -ACL for image segmentation, feature detection/ extraction, image processing, etc.

- Sensor integration (e.g. anomaly detection)
  -M4 manages sensor reading/fusion, feature extraction
  -Then use RPmsg to send data to the A53 for inferencing

Graphics (OpenGL ES

Can be used for color space conversion



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	
Processor and OS Platform	

Combining Horizontal Capabilities to Build Vertical Solutions

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

Processor and OS Platform — freeRTOS, NXP i.MX RT, NXP i.MX 8M Mini

Smart Appliance

Smart Home

Smart Retail

Smart Industry

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

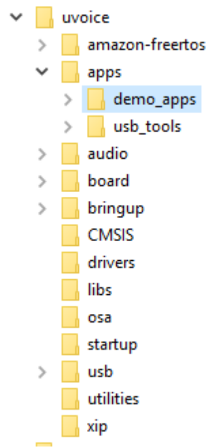
Smart lock	Smart appliance	Smart appliance / Smart Panel
		Secure Facial Recognition
Face Recognition	Anomaly Detection	Anomaly Detection
Voice Control	Voice Control	Voice Control

Production Grade, Certified IoT Edge Machine Learning Solutions

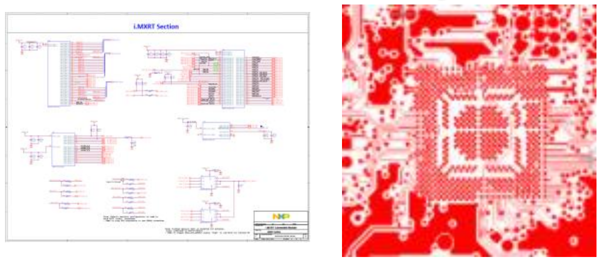
- Implemented with best in class silicon, software and IP from NXP and 3rd parties
- Near production ready hardware
  - Cost and form factor optimized
- Pre-integrated production ready software, fully tested & certified
- NXP provides a single point of contact for support, licensing and procurement
- 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



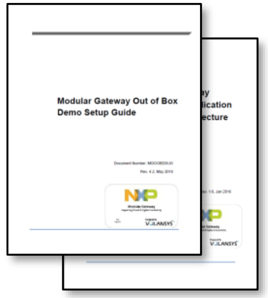
Schematics **Layouts** **Certifications**



BOMs

U1	MIMXRT1052DVL6B	i.MXRT1050 Cross Processor
U3	W9812G6JB-6I	128Mbit SDRAM 3V 166MHz
U4	IS26KL256S-DABLI0(	256Mb Hyperflash 3V 100MHz Fla
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



Thank You!

SECURE CONNECTIONS
FOR A SMARTER WORLD