Machine learning for embedded systems at the edge

Arm and NXP

Kobus Marneweck, Product Manager, Arm
Anthony Huereca, Systems Engineer, NXP

June 16, 2020
<table>
<thead>
<tr>
<th>Date</th>
<th>Title</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Today</td>
<td>Machine learning for embedded systems at the edge</td>
<td>Arm and NXP</td>
</tr>
<tr>
<td>June, 30</td>
<td>tinyML development with Tensorflow Lite for Microcontrollers and CMSIS-NN</td>
<td>Arm</td>
</tr>
<tr>
<td>July, 14</td>
<td>Demystify artificial intelligence on Arm MCUs</td>
<td>Cartesiam.ai</td>
</tr>
<tr>
<td>July, 28</td>
<td>Speech recognition on Arm Cortex-M</td>
<td>Fluent.ai</td>
</tr>
<tr>
<td>August, 11</td>
<td>Getting started with Arm Cortex-M software development and Arm Development Studio</td>
<td>Arm</td>
</tr>
<tr>
<td>August, 25</td>
<td>Efficient ML across Arm from Cortex-M to Web Assembly</td>
<td>Edge Impulse</td>
</tr>
</tbody>
</table>

Visit: developer.arm.com/solutions/machine-learning-on-arm/ai-virtual-tech-talks
SPEAKERS

Kobus Marneweck, Senior Product Manager
Arm

Anthony Huereca, Embedded Systems Engineer
NXP Semiconductor
AGENDA

• ML on the edge
• eIQ deployment
  - Arm support for TFLμ
  - TensorFlow
  - Glow
  - Getting started
• The future
• Wrap-up
Machine Learning on the Edge
EXAMPLE EMBEDDED AI APPLICATIONS

Image Classification
- Identify what camera is looking at
  - Coffee pods
  - Empty vs full trucks
  - Factory defects on manufacturing line
  - Produce on supermarket scale
- Personalization based on facial recognition
  - Appliances
  - Home
  - Toys
  - Auto
- Security Video Analysis

Audio Analysis
- Keyword actions
  - “Alexa”/“Hey Google”
- Voice commands
- Alarm Analytics
  - Breaking glass
  - Crying baby

Anomaly Detection
- Identify factory issues before they become catastrophic
- Smartwatch health monitoring
- Motor performance monitoring
- Sensor Analysis
MACHINE LEARNING PROCESS

1. Training Phase
2. Inference Phase

Training Phase:
- Collect and Prepare Data
- Train Model
- Test Model

Iterate on parameters and algorithm to get best model

Inference Phase:
- Input
- Deployed Model
- Prediction
**INFERENCE ON THE EDGE**

- Inference is using a model to make a prediction on new data
- Data can come from embedded camera, microphone, or sensors

---

**Two possibilities:**

**Inference on the Cloud**
- Requires network bandwidth
- Latency issues
- Cloud compute costs

**Inference on the Edge**
- **Increased privacy and security**
- Faster response time and throughput
- Lower Power
- Don’t need internet connectivity

---

*arm*
NXP Enablement for Machine Learning
NXP BROAD-BASED MACHINE LEARNING SOLUTIONS AND SUPPORT

eIQ™ ML Enablement
- eIQ (edge intelligence) for edge AI/ML inference enablement
- Based on open source technologies (TensorFlow Lite, Arm NN, Glow, ONNX, OpenCV)
- Support for i.MX 8 family, i.MX RT1050/1060/600
- Fully integrated into NXP development environments (MCUXpresso, Yocto/Linux)
- BYOM – Bring Your Own Model

Third Party SW and HW
- Coral Dev Board
- i.MX 8M Development Kit for Amazon® Alexa Voice Service w/ DSP Concepts
- Au-Zone Network Development Tools
- Arcturus video applications
- SensiML tools for sensor analysis

Turnkey Solutions
- Alexa Voice Services (AVS) solution
  - i.MX RT106A (kit – SLN-ALEXA-IOT)
- Local voice control solution
  - i.MX RT106L (kit – SLN-LOCAL-IOT)
- Face & emotion recognition solution
  - i.MX RT106F (kit – SLN-VIZN-IOT)

DIY

Coral

Third Party SW and HW

... And more

SLN-ALEXA-IOT

Fully Tested

.... And more
ARM CORTEX-M PORTFOLIO

- **Cortex-M7**: Maximum performance, control and DSP.
- **Cortex-M4**: Mainstream control and DSP.
- **Cortex-M33**: Flexibility, control and DSP.
- **Cortex-M55**: Helium vector extensions, Optimized for DSP & ML.
- **Cortex-M3**: Lowest cost, low power.
- **Cortex-M0**: Lowest cost, low power.
- **Cortex-M0+**: Highest energy efficiency.
- **Cortex-M23**: Smallest area, lowest power.
- **Cortex-M23**: Flexibility, control and DSP.

**TrustZone**

- **High performance**
- **Performance efficiency**
- **Lowest power & area**

**Cortex-M today**

- Well suited for ML & DSP applications

**Relative control code performance**

**Relative ML and DSP performance**

- Armv6-M
- Armv7-M
- Armv8-M
CORTEX-M7: HIGHEST PERFORMANCE CORTEX-M

High performance – dual-issue processor
- Achieves 2.14 DMIPS/MHz, 5.01 CoreMark/MHz
- Achieves 1.4GHz in 16FFC (typ config with caches and FPU)

Retains all of the Cortex-M benefits
- Ease-of-use, low interrupt latency

Flexible memory interfaces
- Up to 16MB TCM for critical data and code
- Up to 64KB I-cache and D-cache
- AXI master interface

Performance
- Floating-point Unit (FPU) – Single precision (SP) and double precision (DP), sustained 2x 32bit or 2x 16bit MACs per cycle
- Digital signal processing (DSP) extension
CORTEX-M33: NEXT-GENERATION CORTEX-M WITH TRUSTZONE SECURITY

Industry-standard 32bit processor
- 3-stage pipeline, Harvard architecture
- Extremely flexible design configurations

Wide choice of options for differentiated products
- TrustZone security foundation with up to two memory protection units (MPUs)
- Digital signal processing (DSP) extension with SIMD, single-cycle MAC, saturating arithmetic
- Floating-point Unit (FPU)
- Coprocessor interface
- Arm Custom Instructions
- Powerful debug and non-intrusive real-time trace (ETM, MTB)
MACHINE LEARNING PROCESS

Model Frameworks

- TensorFlow
- Keras
- Caffe
- PyTorch
- other...

Training

- Collect and Prepare Data
- Train Model
- Test Model

Optimize (optional)

- Pruning
- Quantization

Convert

- tf lite_convert.py
- code_gen.py
- model_compiler

Inference

- TensorFlow Lite
- CMSIS-NN
- Glow

Note: There is no unified method for converting neural networks from different frameworks to run on Arm Cortex-M products

i.MX RT eIQ inference engine options:

- CMSIS-NN – Can be used for several different model frameworks
- TensorFlow Lite – Used for TensorFlow model frameworks
- Glow – Machine Learning compiler for several different model frameworks (Coming in July)
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
BYOM – Bring Your Own Model

**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**

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 DEMO

• Retrained a Mobilenet model written in TensorFlow to identify 5 different flower types
• Use eIQ to run model on i.MX RT1060 EVK
  − Lab at https://community.nxp.com/docs/DOC-343827
  − Lab steps can be used for any types of images you’re interested in
eIQ Deployment Overview
ADDITIONAL FLAVORS of NXP eIQ™ MACHINE LEARNING DEVELOPMENT ENVIRONMENT

User Application with eIQ Deployment NN Models

COMPUTE ENGINES

<table>
<thead>
<tr>
<th>Cortex-M</th>
<th>DSP</th>
<th>Cortex-A</th>
<th>GPU</th>
<th>ML Accelerator</th>
</tr>
</thead>
<tbody>
<tr>
<td>i.MX RT600</td>
<td>i.MX RT600</td>
<td>i.MX 8M Plus</td>
<td>i.MX 8M Plus</td>
<td>i.MX 8M Plus</td>
</tr>
<tr>
<td>i.MX RT1050</td>
<td>i.MX 8Q0</td>
<td>i.MX 8Q0</td>
<td>i.MX 8Q0</td>
<td>Future MCU</td>
</tr>
<tr>
<td>i.MX RT1060</td>
<td>i.MX 8QP</td>
<td>i.MX 8QP</td>
<td>i.MX 8QP</td>
<td></td>
</tr>
<tr>
<td>i.MX RT1170</td>
<td>i.MX 8QXP</td>
<td>i.MX 8QXP</td>
<td>i.MX 8QXP</td>
<td></td>
</tr>
<tr>
<td>i.MX 8M Quad/Nano</td>
<td>i.MX 8M Mini</td>
<td>i.MX 8M Quad/Nano</td>
<td>i.MX 8M Quad/Nano</td>
<td></td>
</tr>
</tbody>
</table>

NXP EIQ INFERENCE ENGINES & LIBRARIES

TensorFlowLite     GLOW            armNN            ONNX         TensorFlowLite

ML CLOUD TRAINING

MICROSOFT AZURE GOOGLE CLOUD AMAZON WEB SERVICES

TRAINED OPTIMIZED QUANTIZED MODEL

01001 00101
eIQ ADVANTAGES

- eIQ implements performance enhancements with CMSIS-NN for Cortex M cores and DSP
  - Up to 2.4x improvement in inference time in TensorFlow Lite over original code
- eIQ inference engines work out-of-the-box and are already tested and optimized.
  - Get up and running in minutes instead of weeks

NXP eIQ Enablement

Roll Your Own
Inference engines available with eIQ for i.MX RT:

- **CMSIS-NN** – Can be used for several different model frameworks
- **TensorFlow Lite** – Used for TensorFlow model frameworks
- **Glow** – Machine Learning compiler for several different model frameworks (Coming in July)
Arm support for TFLμ
CMSIS-NN INFEERENCE

- 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 models into CMSIS-NN API calls.
- CMSIS-NN optimized the implementation of inference engines like TFLite micro (https://www.tensorflow.org/lite/microcontrollers)
CMSIS-NN OPTIMIZED FOR PERFORMANCE

• Key ML function support
  - Aiming for best-in-class performance for Cortex-M CPUs (compared to other libraries)
  - Available now through open source license
• Consistent interface to all Cortex-M CPUs
  - Extending to Arm v8-M
• Open-source, via Apache 2.0 license
  - https://github.com/ARM-software/CMSIS_5

CMSIS-NN
Optimised for Cortex-M CPUs
Armv7-M Armv8.1-M

CNN Runtime improvement
Series1 Series2

Energy efficiency improvement

4.9x higher eff.

4.6x higher perf.
TOOLS & TFLM OPERATOR SUPPORT – CMSIS-NN AND ETHOS MICRONPU

- TensorFlow Lite
  - Input File
  - Micro TensorFlow Lite runtime
  - Reference kernels
- CMSIS-NN
  - Optimized operators
- Ethos microNPU
  - Driver
- Ethos-U microNPU
- Cortex-M
  - Armv6M Armv7M Armv8M Armv8.1M (MVE)

Optimized custom operators for the microNPU
eIQ TensorFlow
TENSORFLOW LITE INFERENCE ENGINE

• Developed by Google
  - TensorFlow → Training and Inference
  - TensorFlow Lite eIQ → NXP’s implementation of TF Lite for MCUs
  - TensorFlow Lite Micro → TensorFlow’s implementation of TF Lite for MCUs

• Can only be used with TensorFlow models
• Use tflite_convert utility (provided by TensorFlow) to convert a TensorFlow model to a .tflite binary
• TFLite flat buffer binary is read from memory by TFLite inference engine running on i.MX RT
TENSORFLOW LITE CONVERSION PROCESS

1. Transform a TensorFlow .pb model to TFLite flat buffer file.
2. Convert TFLite flat buffer file to C array in a **.h header**
3. Copy .h header file into eIQ TensorFlow Lite SDK example
**TENSORFLOW LITE CODE FLOW**

- **Import model**
  ```cpp
  #include "mobilenet_model.h"
  model = tflite::FlatBufferModel::BuildFromBuffer(mobilenet_model, mobilenet_model_len);
  ```

- **Get input**
  ```cpp
  /* Extract image from camera to data buffer. */
  CSI2Image(data, Rec_w, Rec_h, pExtract, true);
  /* Resize image to input tensor size. */
  ResizeImage(interpreter->tensor(input), data, Rec_h, Rec_w, image_height, image_width, image_channels, &s);
  ```

- **Run inference**
  ```cpp
  interpreter->Invoke();
  ```

- **Get Results**
  ```cpp
  std::vector<std::pair<float, int>> top_results;
  GetTopN<float>(interpreter->typed_output_tensor<float>(0), output_size, s->number_of_results, threshold, &top_results, true);
  auto result = top_results.front(); //Get results
  const float confidence = result.first;  //Get confidence level
  const int index = result.second;   //Get highest class
  ```
GEMMLowP Assembly-Coded DSP Optimization Benefits for TensorFlow Lite

**GCC Arm® 8-2018-q4**

<table>
<thead>
<tr>
<th></th>
<th>DSP Optimized (-O2)</th>
<th>Reference Kernel (-O2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label Image</td>
<td>186 ms</td>
<td>370 ms</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>61 ms</td>
<td>229 ms</td>
</tr>
</tbody>
</table>

**IAR EW 8.32.3**

<table>
<thead>
<tr>
<th></th>
<th>DSP Optimized</th>
<th>Original</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label Image</td>
<td>217 ms</td>
<td>307 ms</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>67 ms</td>
<td>159 ms</td>
</tr>
</tbody>
</table>

**Keil MDK 5.27**

<table>
<thead>
<tr>
<th></th>
<th>DSP Optimized</th>
<th>Original</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label Image</td>
<td>178 ms</td>
<td>198 ms</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>64 ms</td>
<td>87 ms</td>
</tr>
</tbody>
</table>
eIQ Glow
Developed by Facebook

Glow is a compiler that turns a model into a machine executable binary for the target device
- Both the model and the inference engine are compiled into the binary that is generated.
- Integrate the generated binary into an SDK software project
- Can make use of compiler optimizations
- Supports ONNX (universal model format) and Caffe2 models

Cutting-edge inference technology
PERFORMANCE COMPARISON USING CIFAR-10 MODEL ON RT1050
OPTIMIZATIONS FOR GLOW

• NXP developed optimizations for Glow on i.MX RT devices
• Operations can be dispatched to the HiFi4 DSP on RT685
  - HiFi4 DSP increases performance up to 34x
• Operations can also use CMSIS-NN library optimizations for all Glow supported devices
1. Transform model to the universal ONNX format.
2. Optimize model with profiler to create profile.yml file for quantization
3. Compile with Glow model_compiler to generate compiled files and weights.
4. Copy binary files into eIQ Glow SDK example.
ADD COMPILLED CODE TO PROJECT

• Add <network_name>.o compiled file to project settings
• Include <network_name>.h file
• Set input data

```c
// Load input data
memcpy(bundleInpAddr, ((char *)INPUT_DATA_START) + idx * INPUT_IMAGE_SIZE, INPUT_IMAGE_SIZE);
```

• Run model

```c
// Perform inference and compute inference time.
start_time = get_time_in_us();
mlist(constantWeight, mutableweight, activations);
stop_time = get_time_in_us();
duration_ms = (stop_time - start_time) / 1000;
```

• Get result

```c
// Get classification top1 result and confidence
float "out_data" = (float *)bundleOutAddr;
float max_val = 0.0;
uint32_t max_idx = 0;
for (int i = 0; i < OUTPUT_NUM_CLASS; i++)
{
    if (out_data[i] > max_val)
    {
        max_val = out_data[i];
        max_idx = i;
    }
}
GLOW MEMORY USAGE

• Glow does not use dynamically allocated memory (heap).
• All the memory requirements of a compiled model can be found in the auto-generated header file.

    // Memory sizes (bytes).
    #define CIFAR10_CONSTANT_MEM_SIZE     34176   // Stores model weights. Can be stored in Flash or RAM.
    #define CIFAR10_MUTABLE_MEM_SIZE      12352   // Stores model inputs/outputs. Must be in RAM.
    #define CIFAR10_ACTIVATIONS_MEM_SIZE  71680   // Store model scratch memory required for intermediate computations. Must be in RAM.
Getting elQ
eIQ IN MCUXPRESSO SDK

- eIQ for the i.MX RT family is included as part of MCUXpresso SDK (https://mcuxpresso.nxp.com/en/welcome)

- Make sure eIQ selected in MCUXpresso SDK builder:
eIQ EXAMPLES

eIQ RT1060 SDK examples available:

<table>
<thead>
<tr>
<th>Description</th>
<th>CIFAR-10</th>
<th>Keyword Spotting (KWS)</th>
<th>Label Image</th>
<th>Anomaly Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Classifies 32x32 image from camera input into one of 10 categories</td>
<td>Detects specific keywords from microphone input</td>
<td>Classifies 128x128 image from camera input into one of 1000 categories using Mobilenet model</td>
<td>Use FRDM-STBC-AGM01 sensor board for accelerometer anomaly analysis (Select “agm01” board)</td>
</tr>
<tr>
<td>TensorFlow Lite Example</td>
<td><img src="image1" alt="airplane" /> <img src="image2" alt="automobile" /> <img src="image3" alt="bird" /> <img src="image4" alt="cat" /> <img src="image5" alt="dear" /> <img src="image6" alt="dog" /> <img src="image7" alt="frog" /> <img src="image8" alt="horse" /></td>
<td><img src="image9" alt="Input speech signal" /></td>
<td><img src="image10" alt="TensorFlow Lite Example" /></td>
<td><img src="image11" alt="Anomaly Detection" /></td>
</tr>
</tbody>
</table>
eIQ FOLDER STRUCTURE

- **Project Files for Examples**
- **Project Files for TensorFlow Lite Library**
- **Source Code for CMSIS-NN Examples**
- **CMSIS-NN Source Code**
- **Source Code for TensorFlow Lite Examples**
- **TensorFlow Lite Source Code**
• Anomaly Detection with eIQ using K-Means clustering in TF-Lite (AN12766)
• Handwritten Digit Recognition using TensorFlow Lite (AN12603)

Coming Soon:
• Transfer Learning and Datasets
INFERENCER TIMES

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

• Inference time heavily dependent on the particular model
  - Different images (if same size) will not affect inference time

• Each eIQ example reports inference time

<table>
<thead>
<tr>
<th>Image Classification (ms)</th>
<th>CIFAR-10 (32x32 input)</th>
<th>Mobilenet (128x128 input)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT685 w/ HiFi4 Glow</td>
<td>6.7</td>
<td>61</td>
</tr>
<tr>
<td>RT1060 Glow</td>
<td>24</td>
<td>74</td>
</tr>
<tr>
<td>RT1060 TensorFlow Lite</td>
<td>64</td>
<td>178</td>
</tr>
</tbody>
</table>
MEMORY REQUIREMENTS

- Flash: Model, inference engine code, and input data
- RAM: Intermediate products of the model layers
  - Size depends on amount of data, size, and type of the layers and is very model dependent

Benchmark and optimizations ongoing. Numbers subject to change:

<table>
<thead>
<tr>
<th>Model</th>
<th>Inference Engine</th>
<th>Flash</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>CMSIS-NN</td>
<td>110KB</td>
<td>50KB</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>TensorFlow Lite</td>
<td>600KB (92KB model, 450KB engine)</td>
<td>320KB</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>Glow</td>
<td>69KB</td>
<td>131KB</td>
</tr>
<tr>
<td>Mobilenet v1</td>
<td>TensorFlow Lite</td>
<td>1.5MB (450KB model, 450KB engine)</td>
<td>2.5MB</td>
</tr>
<tr>
<td>Mobilenet v1</td>
<td>Glow</td>
<td>507KB</td>
<td>1MB</td>
</tr>
</tbody>
</table>
The future
PUSHING THE BOUNDARIES FOR REAL-TIME ON-DEVICE PROCESSING

Cortex-M today

Cortex-M7
Cortex-M35P
Cortex-M33
Cortex-M3
Cortex-M23
Cortex-M0+
Cortex-M0
Cortex-M1

New Cortex-M CPU enabled by Helium

Cortex-M55

Arm microNPUs

Cortex-M55 + Ethos-U55 (multiple performance points available)

Signal conditioning and ML foundation

ML performance and efficiency

Relative ML and DSP performance

Relative control code performance

Well suited for ML & DSP applications
CORTEX-M55 & ETHOS-U55: TRANSFORMING CAPABILITIES OF THE SMALLEST DEVICES

Boosting signal processing and ML performance for millions of developers

Signal processing → Machine learning

Signal conditioning → Feature extraction → Decision algorithm

Cortex-M55
Up to **5x** higher signal processing performance (CFFT in int32)

Cortex-M55
Up to **15x** higher ML performance* (matrix multiplication in int8)

Cortex-M55 & Ethos-U55
Up to **480x** higher ML performance* (matrix multiplication in int8)

*Compared to existing Armv8-M implementations
Summary
FURTHER READING

• NXP eIQ
• TensorFlow Lite
• Glow
• 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 Practical
• YouTube series on the basics of ML and TensorFlow (ML Zero to Hero Series)

Book:
You Look Like a Thing and I Love You: How Artificial Intelligence Works and Why It's Making the World a Weirder Place
GIT REPOS

- TensorFlow Lite
- TensorFlow Lite for Microcontrollers
  - https://www.tensorflow.org/lite/microcontrollers

- 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 i.MX RT is included in MCUXpresso SDK
  - [https://mcuxpresso.nxp.com](https://mcuxpresso.nxp.com)
  - TF-Lite and CMSIS-NN eIQ User Guides in SDK documents

- eIQ available for i.MX RT1050 and i.MX RT1060 today
  - Can also run on i.MX RT1064: [https://community.nxp.com/docs/DOC-344225](https://community.nxp.com/docs/DOC-344225)

- eIQ available for i.MX RT685 in July

- Transfer Learning Lab: [https://community.nxp.com/docs/DOC-343827](https://community.nxp.com/docs/DOC-343827)
Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה
Join our next virtual tech talk: tinyML development with Tensorflow Lite for Microcontrollers and CMSIS-NN

Tuesday 30 June

Register here: developer.arm.com/solutions/machine-learning-on-arm/ai-virtual-tech-talks
Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה