tinyML® Talks

Enabling Ultra-low Power Machine Learning at the Edge

"TinyML FPGA implementation for condition monitoring"

Altaf Khan and Martin Kellermann– Infxl LLC and Microchip Technology GmbH

May 12, 2021

www.tinyML.org

tinyML Talks Sponsors

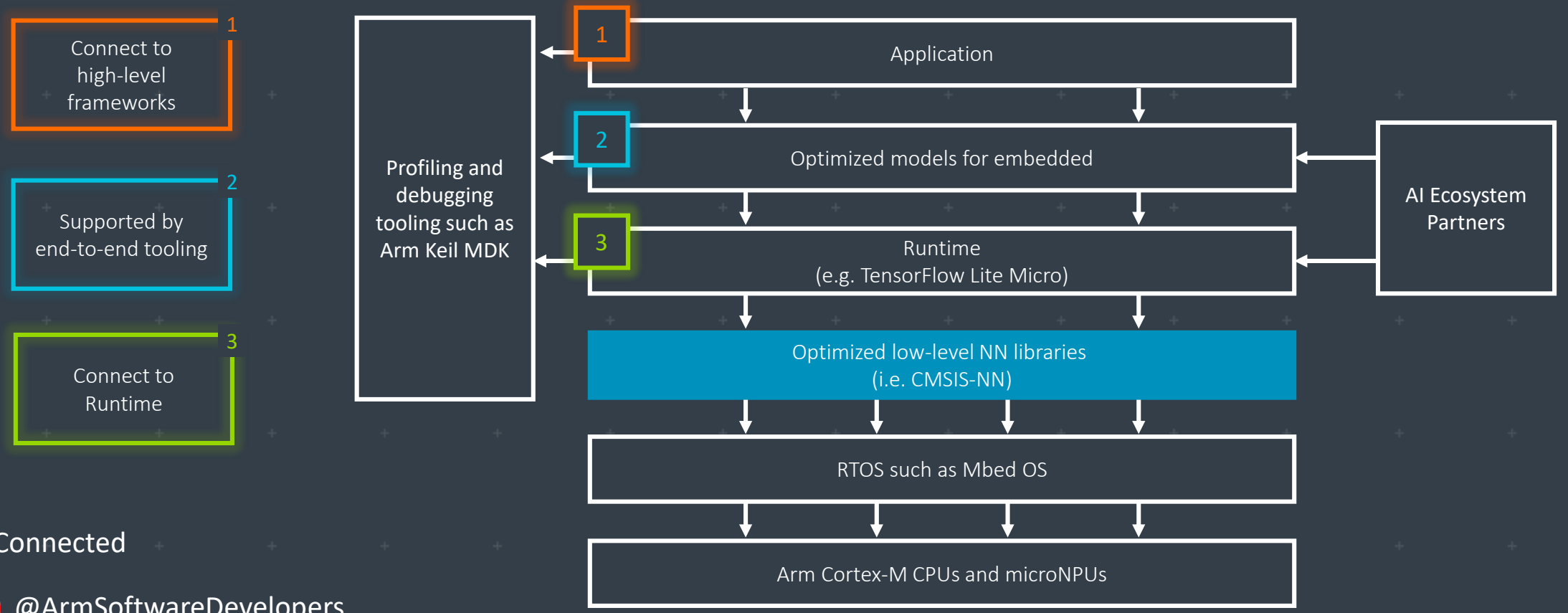
tinyML Strategic Partner

tinyML Strategic Partner

tinyML Strategic Partner

Additional Sponsorships available – contact Olga@tinyML.org for info

Arm: The Software and Hardware Foundation for tinyML



Stay Connected

@ArmSoftwareDevelopers

@ArmSoftwareDev

Resources: developer.arm.com/solutions/machine-learning-on-arm

arm

Qualcomm
AI research

Advancing AI research to make efficient AI ubiquitous

Power efficiency

Model design, compression, quantization, algorithms, efficient hardware, software tool

Personalization

Continuous learning, contextual, always-on, privacy-preserved, distributed learning

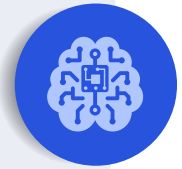
Efficient learning

Robust learning through minimal data, unsupervised learning, on-device learning

A platform to scale AI across the industry

Perception

Object detection, speech recognition, contextual fusion



Reasoning

Scene understanding, language understanding, behavior prediction

Action

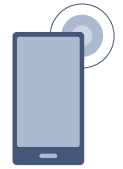
Reinforcement learning for decision making

Edge cloud

Cloud

IoT/IIoT

Automotive



Mobile

Deeplite

WE USE AI TO MAKE OTHER AI FASTER, SMALLER AND MORE POWER EFFICIENT

Automatically compress SOTA models like MobileNet to <200KB with **little to no drop in accuracy** for inference on resource-limited MCUs

Reduce model optimization trial & error from weeks to days using Deeplite's **design space exploration**

Deploy more models to your device without sacrificing performance or battery life with our **easy-to-use software**

BECOME BETA USER **bit.ly/testdeeplite**

mobilityXlab arm AI 100 CBINSIGHTS

TinyML for all developers

C++ library

Arduino library

WebAssembly

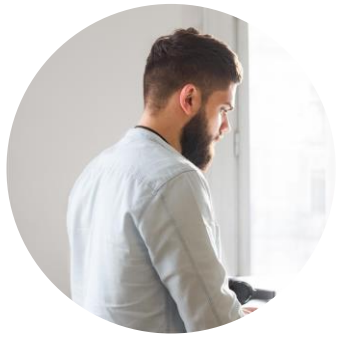
Dataset

Acquire valuable training data securely

Enrich data and train ML algorithms

Edge Device

Real sensors in real time
Open source SDK

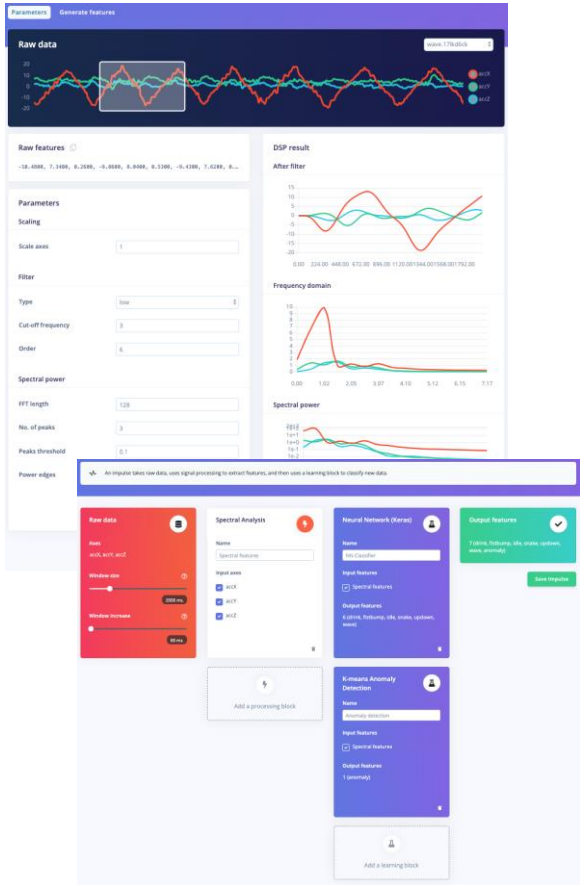


Impulse

Embedded and edge compute deployment options

Test impulse with real-time device data flows

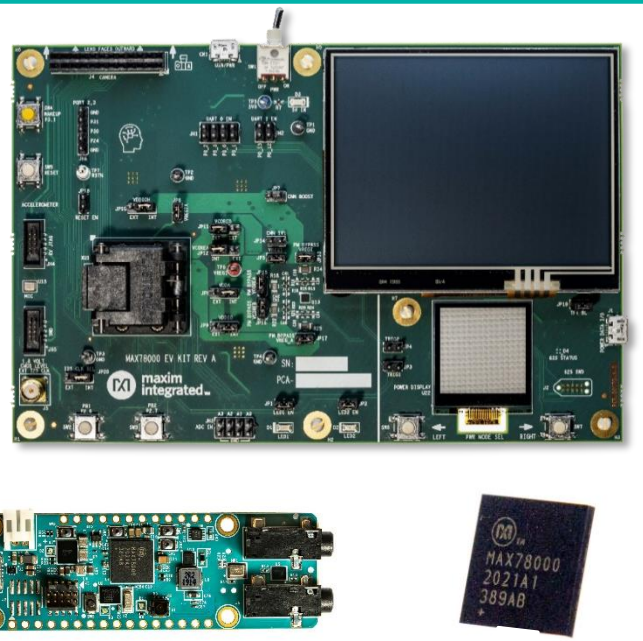
Test



www.edgeimpulse.com

Maxim Integrated: Enabling Edge Intelligence

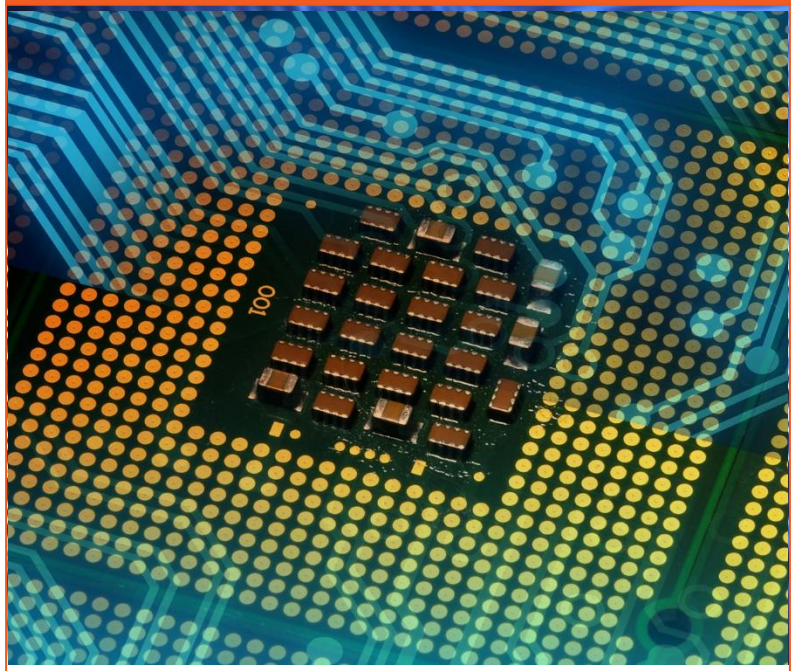
Advanced AI Acceleration IC



The new MAX78000 implements AI inferences at low energy levels, enabling complex audio and video inferencing to run on small batteries. Now the edge can see and hear like never before.

www.maximintegrated.com/MAX78000

Low Power Cortex M4 Micros



Large (3MB flash + 1MB SRAM) and small (256KB flash + 96KB SRAM, 1.6mm x 1.6mm) Cortex M4 microcontrollers enable algorithms and neural networks to run at wearable power levels.

www.maximintegrated.com/microcontrollers

Sensors and Signal Conditioning

Health sensors measure PPG and ECG signals critical to understanding vital signs. Signal chain products enable measuring even the most sensitive signals.

www.maximintegrated.com/sensors

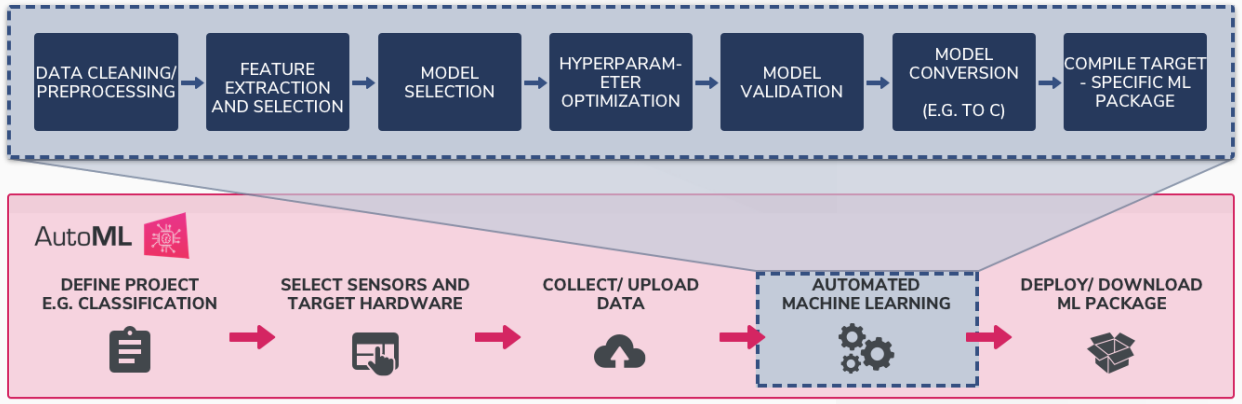
Qeexo AutoML

Automated Machine Learning Platform that builds tinyML solutions for the Edge using sensor data

Key Features

- Supports 17 ML methods:
  - Multi-class algorithms: GBM, XGBoost, Random Forest, Logistic Regression, Gaussian Naive Bayes, Decision Tree, Polynomial SVM, RBF SVM, SVM, CNN, RNN, CRNN, ANN
  - Single-class algorithms: Local Outlier Factor, One Class SVM, One Class Random Forest, Isolation Forest
- Labels, records, validates, and visualizes time-series sensor data
- On-device inference optimized for low latency, low power consumption, and small memory footprint applications
- Supports Arm® Cortex™- M0 to M4 class MCUs

End-to-End Machine Learning Platform



For more information, visit: www.qeexo.com

Target Markets/Applications

- Industrial Predictive Maintenance
- Smart Home
- Wearables
- Automotive
- Mobile
- IoT

Reality AI®

Add Advanced Sensing to your Product with Edge AI / TinyML

https://reality.ai info@reality.ai @SensorAI Reality AI

Pre-built Edge AI sensing modules, plus tools to build your own

Reality AI solutions

Prebuilt sound recognition models for indoor and outdoor use cases

Solution for industrial anomaly detection

Pre-built automotive solution that lets cars "see with sound"

Reality AI Tools® software

Build prototypes, then turn them into real products

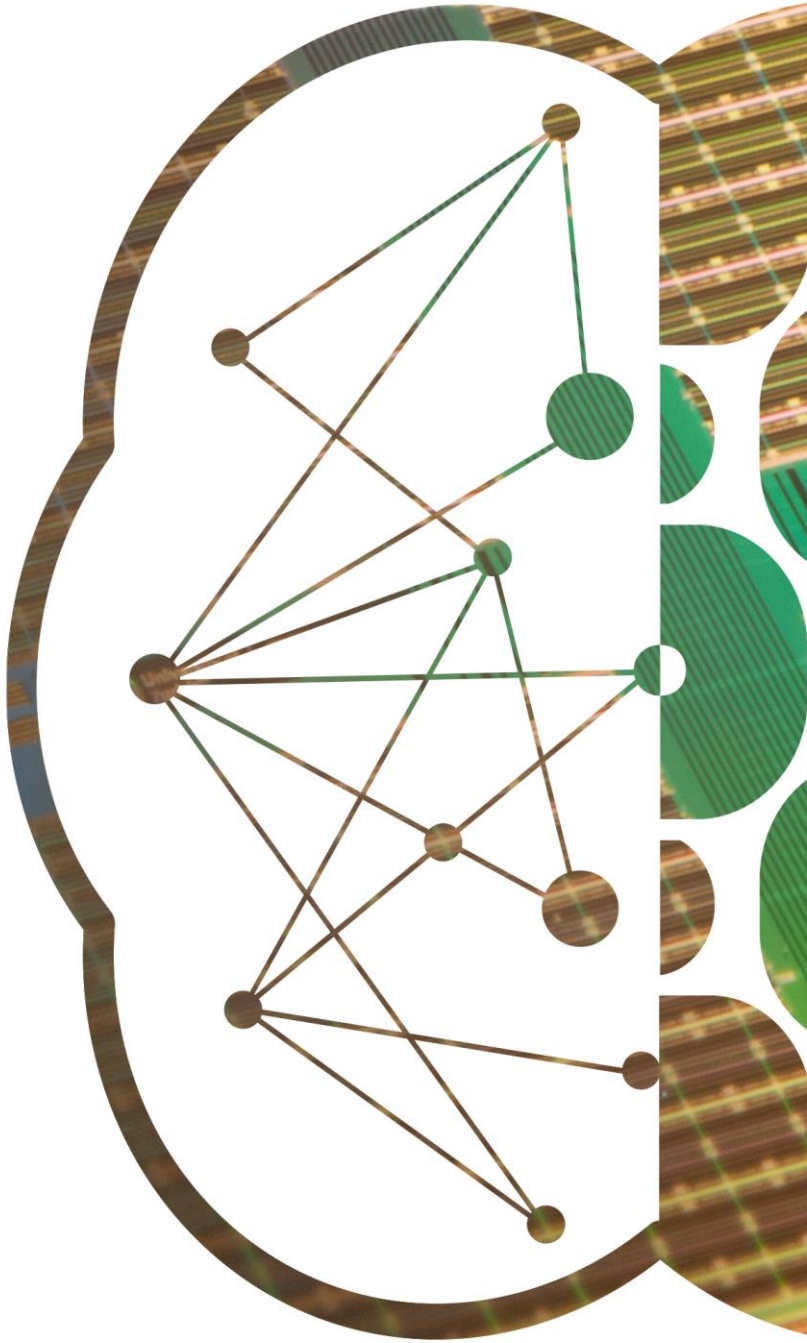
Explain ML models and relate the function to the physics

Optimize the hardware, including sensor selection and placement

SynSense

SynSense builds **sensing and inference** hardware for **ultra-low-power** (sub-mW) **embedded, mobile and edge** devices. We design systems for **real-time always-on smart sensing**, for audio, vision, IMUs, bio-signals and more.

https://SynSense.ai



TINY ML FOUNDATION

collaboration with hackster.io

AN AVNET COMMUNITY

Focus on:
(i) developing new use cases/apps for tinyML vision; and (ii) promoting tinyML tech & companies in the developer community

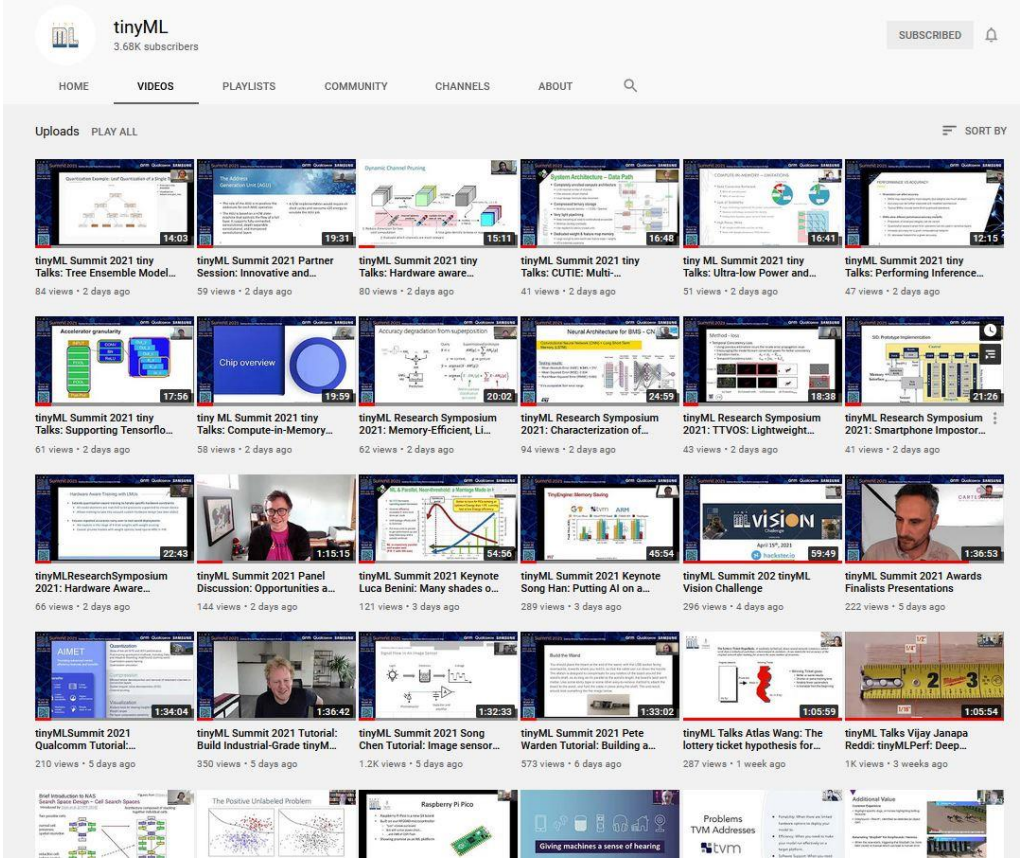
Submissions accepted until August 15th, 2021
Winners announced on September 1, 2021 ($6k value)
Sponsorships available: *sponsorships@tinyML.org*

https://www.hackster.io/contests/tinyml-vision

Successful tinyML Summit 2021:

- **5** days of tutorials, talks, panels, breakouts, symposium
  - **4** tutorials
  - **6** keynotes & **6** plenary tinyTalks (more in breakouts)
  - **2** panel discussions
  - **5** disruptive news presentations
  - **17** breakout/partner sessions
  - **6** Best Product and Innovation Award Finalists & Presentations
  - **89** Speakers
- **5006** registered attendees representing:
  - **104** countries, **1000+** companies and **400+** academic institutions
- **26** Sponsoring companies

www.youtube.com/tinyML with 150+ videos



tinyML Summit-2022, January 24-26, Silicon Valley, CA

EMEA

June 7-10, 2021 (virtual, but LIVE)

Deadline for abstracts: May 1

Sponsorships are being accepted: sponsorships@tinyML.org

Next tinyML Talks

Date	Presenter	Topic / Title
Tuesday, May 25	**Kartik Thakore,** Cofounder, HOTG	Building TinyML Applications Using Rune

Webcast start time is 8 am Pacific time

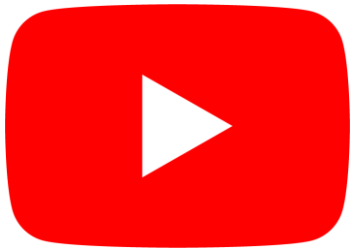
Please contact talks@tinyml.org if you are interested in presenting

TINY ML TALKS webcast

Reminders

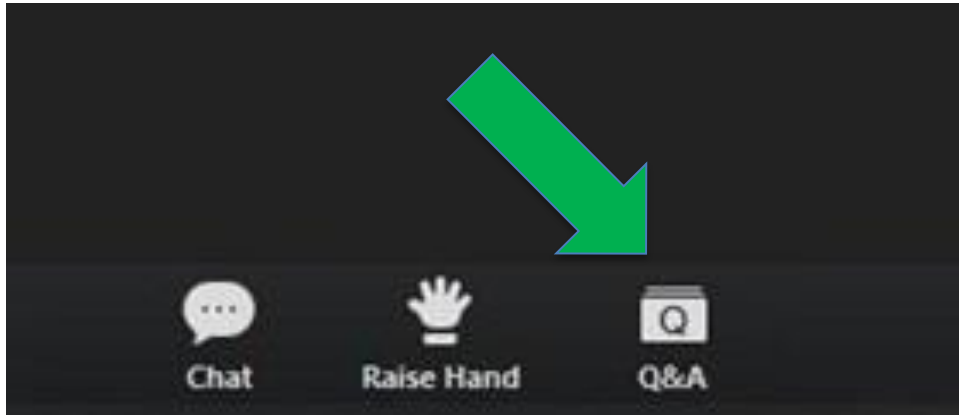
Slides & Videos will be posted tomorrow

tinyml.org/forums



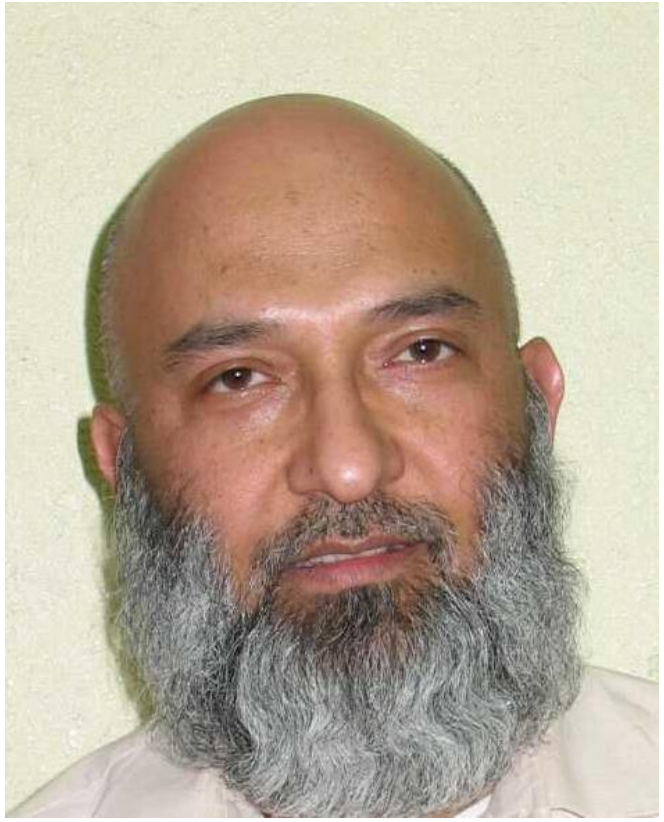
youtube.com/tinyml

Please use the Q&A window for your questions



TINY ML TALKS webcast

Altaf Khan



Altaf Khan is the CEO of Infxl LLC, Colleyville, TX. He started his career as an accelerometer system engineer in Silicon Valley, but simplifying neural nets has been his passion over the last three decades. He has developed fast deep nets for real-time applications, low-cost deep nets for battery-operated IoT endpoints, and small-footprint deep nets for FPGA. He has developed intelligent solutions for a major US airline and a well-known auto parts supplier. He has been the CTO of a brokerage company, CEO of two startups, consultant for software process improvement, and an industrial controls engineer. Altaf received his BSEE from Wilkes College, MSEE for the University of Pennsylvania, and PhD from the University of Warwick.

TINY ML TALKS webcast

Martin Kellermann

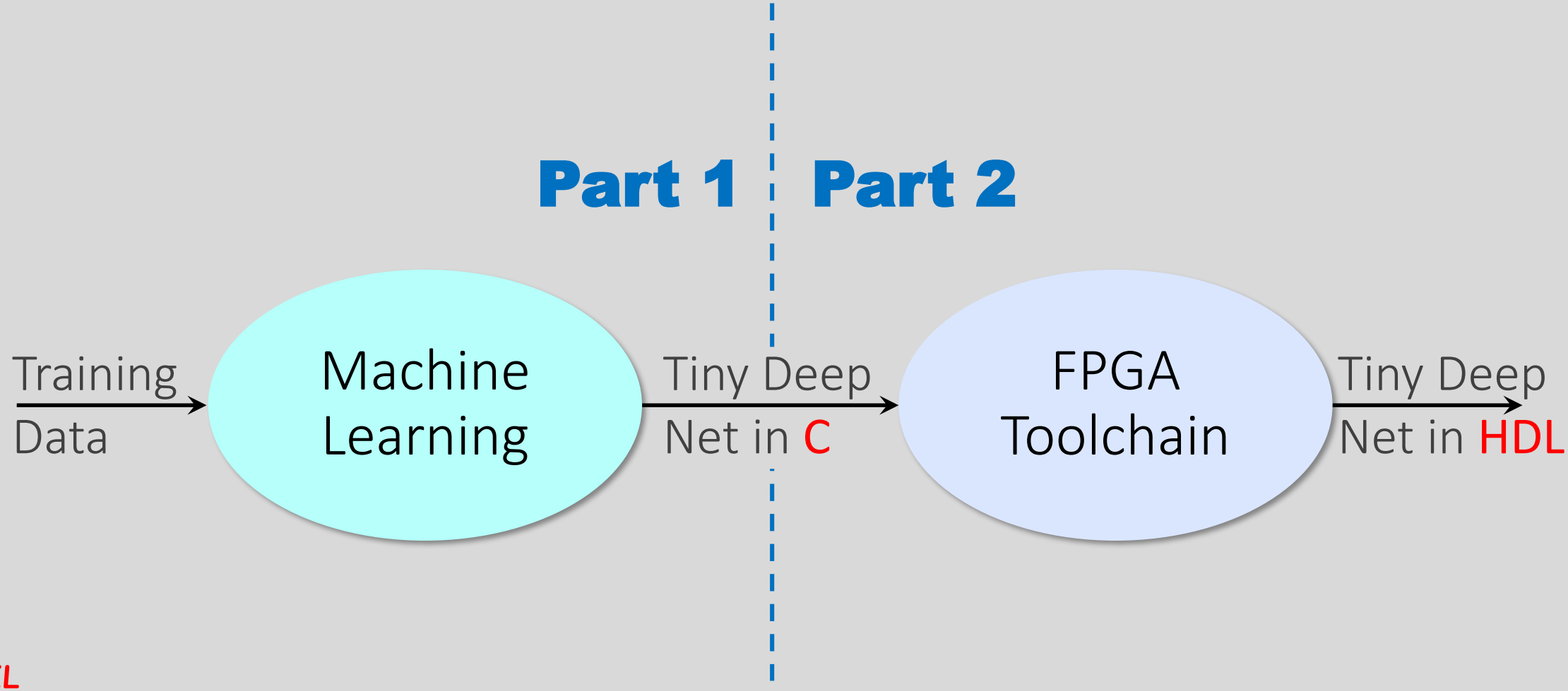
Martin Kellermann is a Marketing Manager at Microchip Technology GmbH, Munich. Earlier he was a Staff Field Application Engineer at Xilinx. He is a seasoned FPGA and SoC professional with a track record of successful customer and project engagements in the industrial, automotive, and data-center domains. He possesses a strong background in high-speed serial data transmission, signal integrity, and hardware debugging which helped numerous customers finish their designs successfully. He has also taught courses covering industrial applications and hardware concepts. Martin is a graduate of the Landshut University of Applied Sciences.

TinyML FPGA Implementation for Condition Monitoring

Part 1

ALTAF KHAN
ALTAF@INFXL.COM

Two-Part Presentation



1. Main benefit of using FPGA for ML inference
2. Main benefit of using FPGA for ML inference in IoT edge devices
3. Main reason why you are hesitant to use FPGAs

FPGA Advantages

- Low latency
  - Parallel compute elements
- Low power
  - No unnecessary compute elements
- Field re-programmable

Use Case: **Condition Monitoring**

Infer the state of a fan based on sensor data

STATES

1. Normal
2. Low voltage
3. Stuck object
4. Obstructed

SENSORS

1. Triaxial Accelerometer
2. PWM
3. Tachometer

Training Data

4,000,004 exemplars, equally distributed over the 4 states

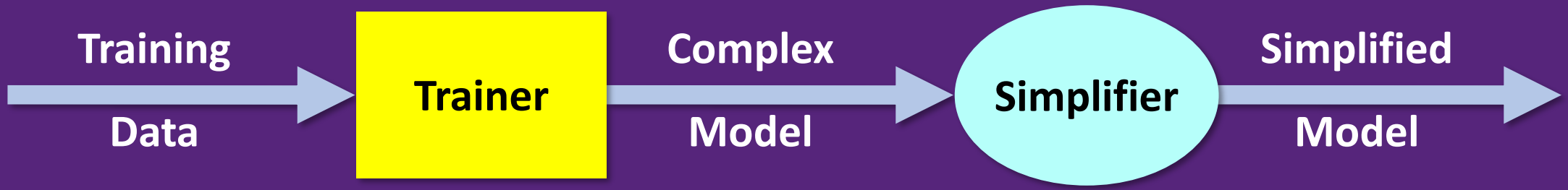
183 duplicate exemplars were removed

1,199,989 (30%) were randomly selected & assigned to test subset

839,680 (30%) of the *remaining* assigned to validation subset

1,959,252 *remaining* exemplars assigned to training subset

Train, and then simplify



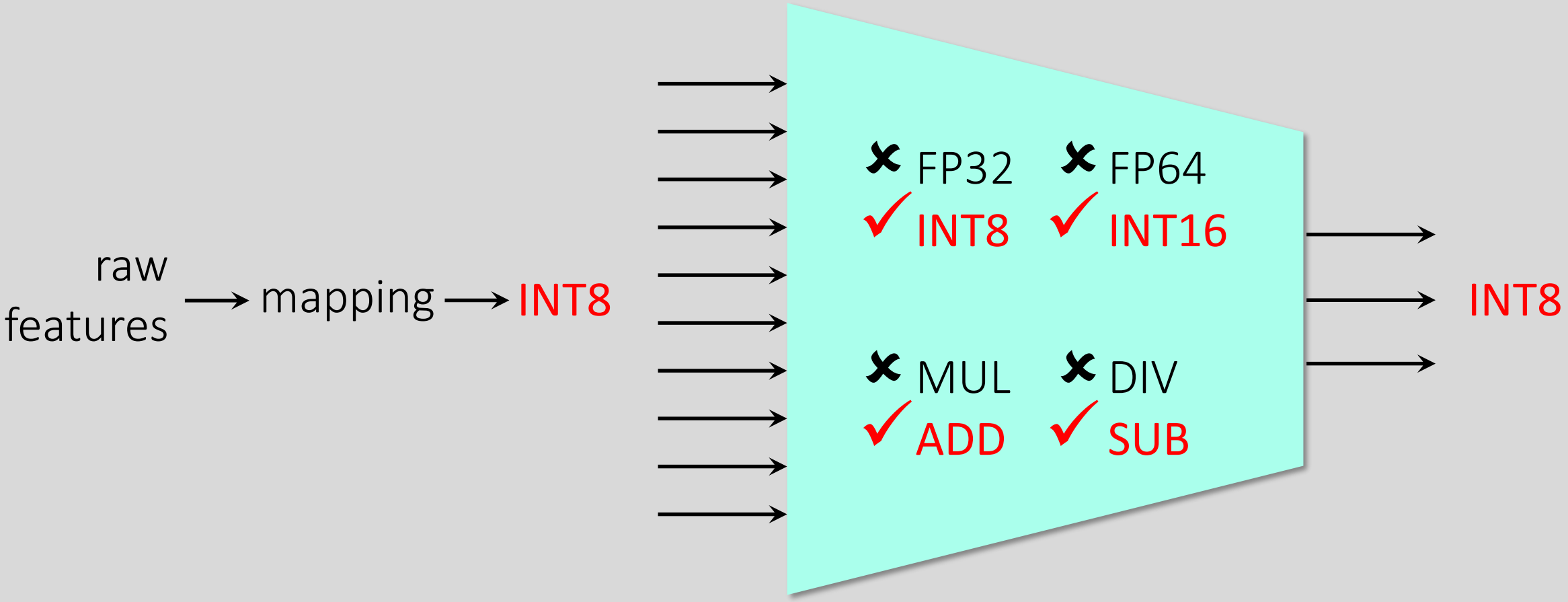
Train and simplify, simultaneously

Infxl Net in C

```
int16_t const ROM16[] = { 5, 4, 130, -12, 42, 4, 21, 22, 27, 35, 57… };
int8_t ram8[ *ROM16 + *(ROM16 + 2) ];

void deep_net_inference(int8_t *ram8) {
   int8_t  const static ROM8[] = { -126, -126, -126, -126, -126… };
   int16_t const *ROM16_ptr = ROM16 + 2;
   int8_t  *nr_ptr = ram8 + *ROM16 - 1;
   int16_t acc16;

   while (   *++ROM16_ptr != -32768) { acc16  =           *ROM16_ptr;
      while (*++ROM16_ptr != -32768) { acc16 += *(ram8 + *ROM16_ptr);
      while (*++ROM16_ptr != -32768) { acc16 -= *(ram8 + *ROM16_ptr);

      if (     acc16 <    0) { *++nr_ptr = -127; }
      else if (acc16 >  656) { *++nr_ptr =  127; }
      else {                   *++nr_ptr = *(acc16 + ROM8); } } }
```

- This code is universal. What differs from project to project are the contents of **ROM16**
- Deep net inputs are written to **ram8** head and the outputs are read from **ram8** tail
- **ROM8** is 657 B. **ROM16** is 1,800 B & **ram8** is 137 B for the Fan State Inference task

Infxl Net: Simple Data, Simple Ops



Infxl Net: I/O Mapping

	I/O Type	I/O Range
Conventional Net	FP32	[0, 1] or [–1, 1]
Infxl Net	INT8	[–127, 127]

Mapping

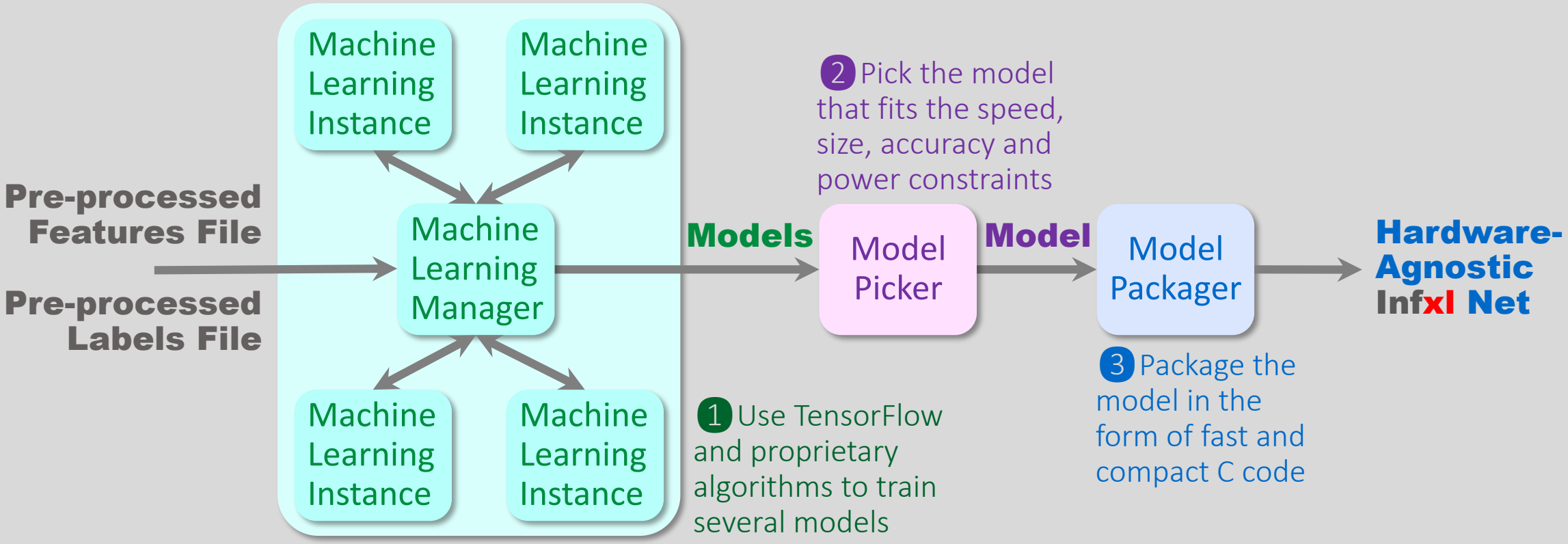
$$[-1, 1] \longrightarrow \texttt{round(127 * x)}$$
$$[0, 1] \longrightarrow \texttt{round(254 * x) - 127}$$

Preprocessing Parameters

Feature	Transform	Offset	Gain
0	Robust	-180	0.92701
1	Standard	-16242	2.97320
2	Robust	-73	0.49416
3	Standard	5669	0.07610
4	Standard	1250	0.32836

$$x \leftarrow \text{clip}[\ (x - offset) * gain\ ]_{-1,1}$$

Infxl Net: Training Workflow



Sensor Examples

Infxl net is optimized for non-vision, non-voice data originating from a variety of sensors

- Accelerometer
- Gyroscope
- Vibration
- Touch

- Temperature
- Microphone
- Pressure
- Flow

- Biological
- Chemical
- Moisture
- Electrical

Fan-State Results on Test Subset

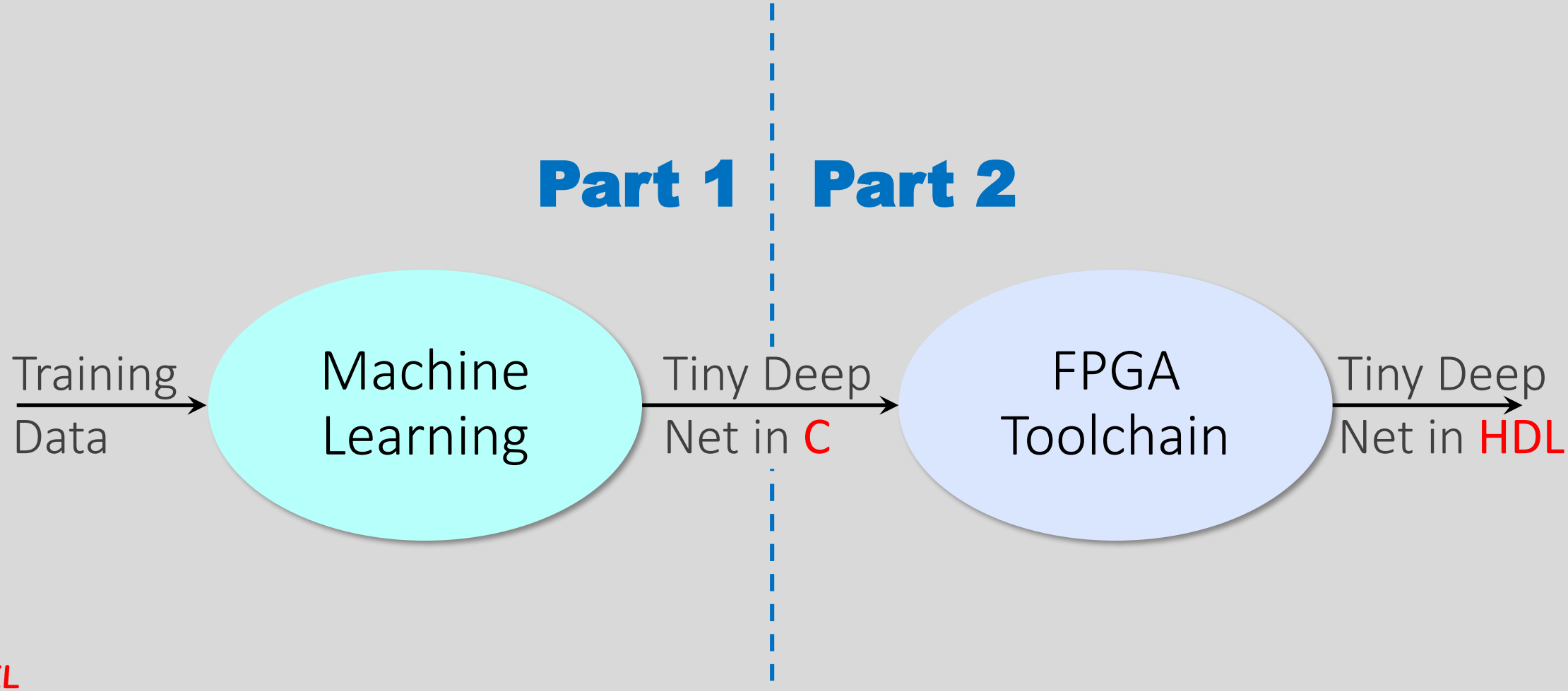
State	Precision	Recall	F1-Score	Support
Normal	0.98	0.98	0.98	300,018
Low Voltage	0.94	0.94	0.94	300,012
Object Stuck	0.93	0.94	0.93	299,886
Obstruction	0.99	0.99	0.99	300,073

Key Observations: Infxl Net as FPGA

- Small size and low power consumption because:
  1. All data paths are either 8 or 16-bit wide
  2. No floating-point operations
  3. No multipliers

- The only difference among projects is a single vector in the C code
  - Once Infxl Net's HDL is optimized for a project on a given FPGA family, doing the next project requires minimal engineering effort

End of Part 1


Part 1
Part 2
Training Data
Machine Learning
Tiny Deep Net in C
FPGA Toolchain
Tiny Deep Net in HDL

TinyML FPGA Implementation for Condition Monitoring

A Leading Provider of Smart, Connected and Secure Embedded Control Solutions

Martin Kellermann
https://www.linkedin.com/in/martinkellermann
Martin.Kellermann@microchip.com

12.05.2021

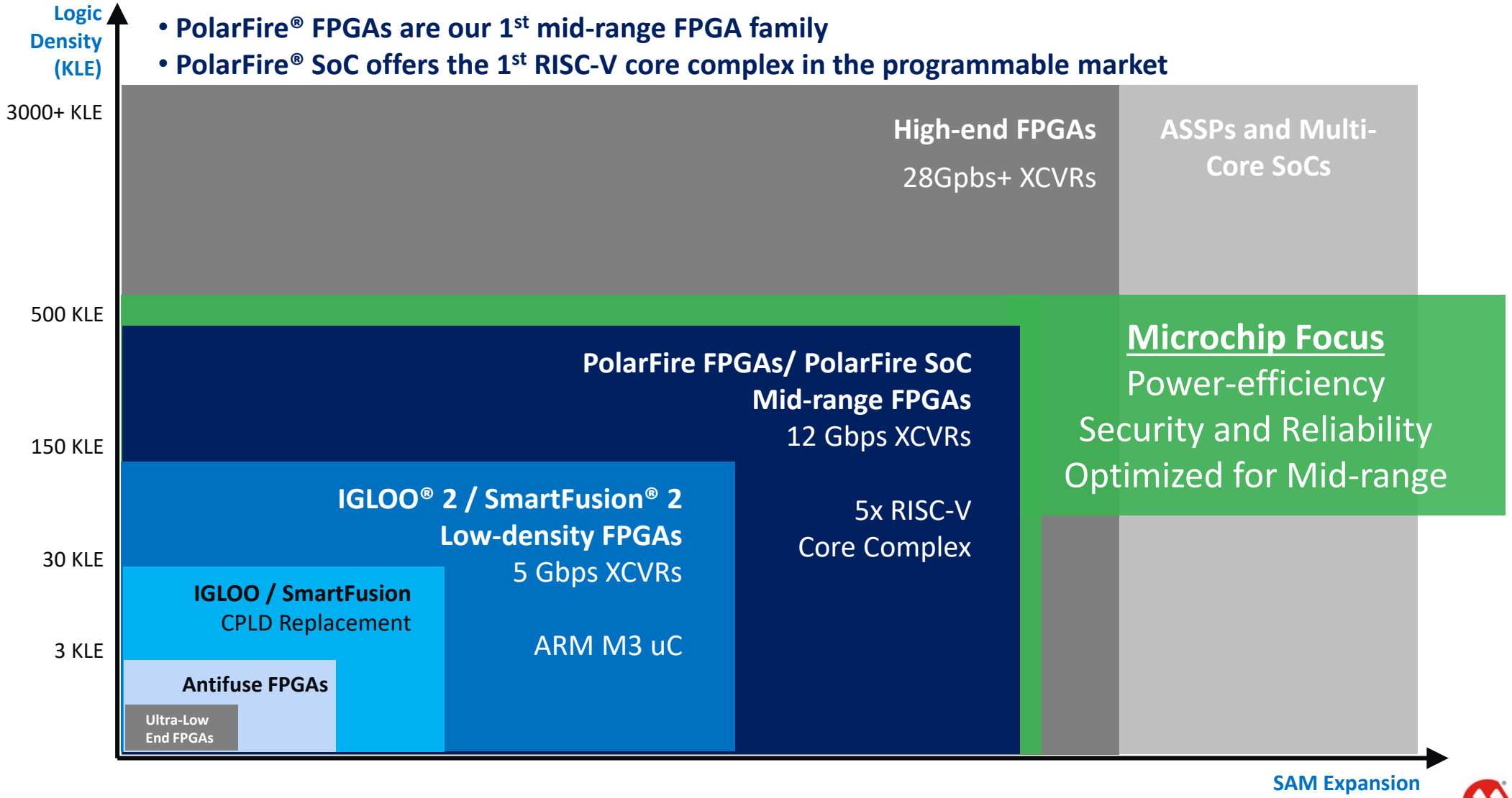
Background

Microchip FPGAs – Made for Low Power

Microchip FPGA / SoC Focus

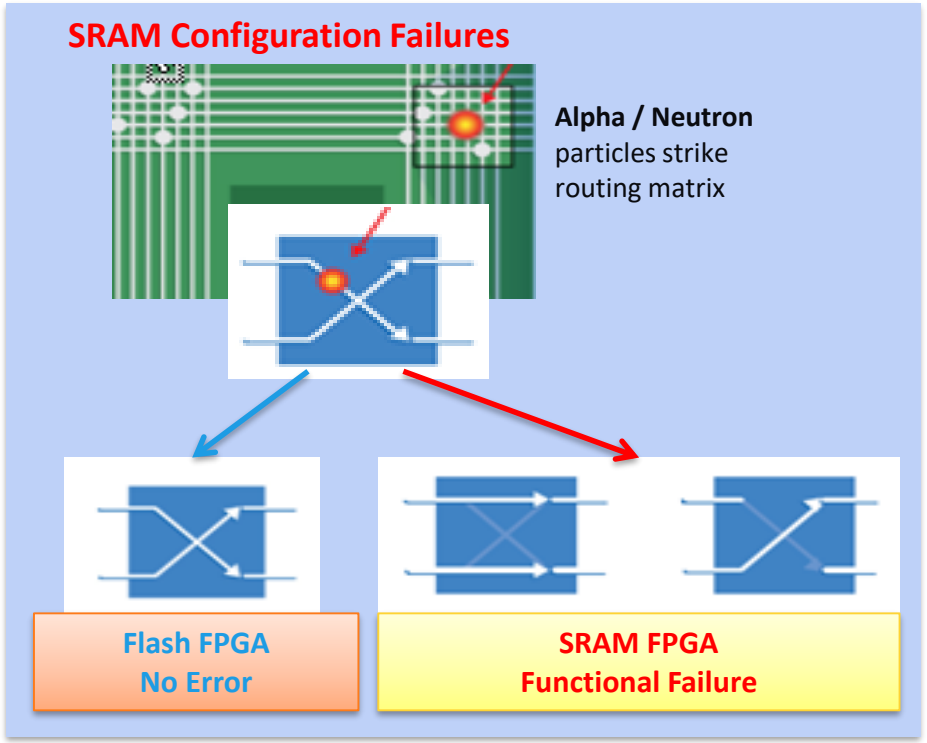
Low Power, Security and Reliability

- PolarFire® FPGAs are our 1st mid-range FPGA family
- PolarFire® SoC offers the 1st RISC-V core complex in the programmable market



Exceptional Reliability

- **Error Free SEU Immune Fabric Configuration**
  - No need to detect configuration errors
    - No scrubbing required
    - No triple mode redundancy needed
    - Lowers cost
    - 24/7/365 availability
- **Block RAM with ECC**
  - Built-in SECDED on 33-bit word
- **System Controller Suspend Mode for Safety Critical Applications**



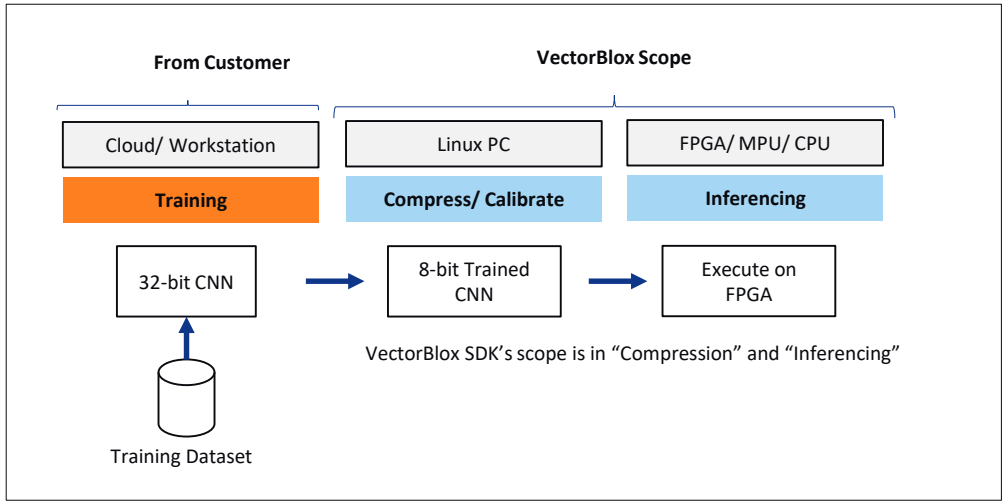
SEU immune fabric simplifies your design and increases reliability

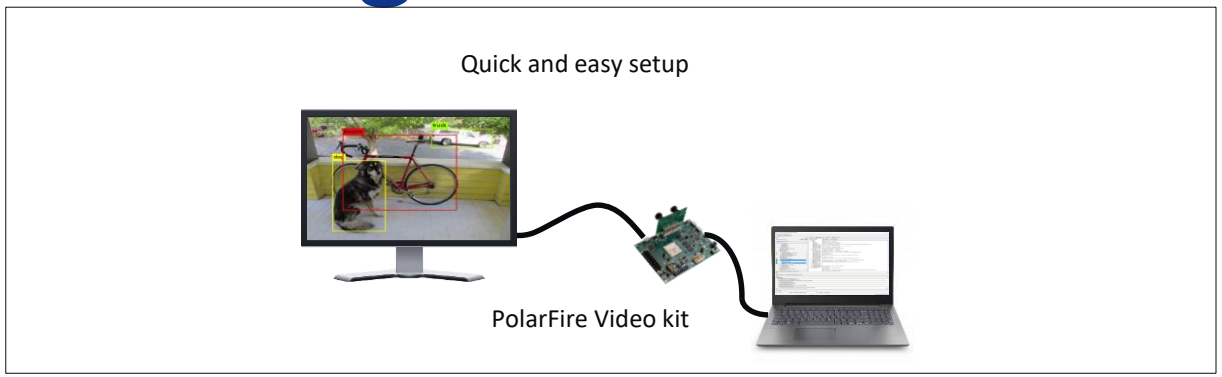
MICROCHIP

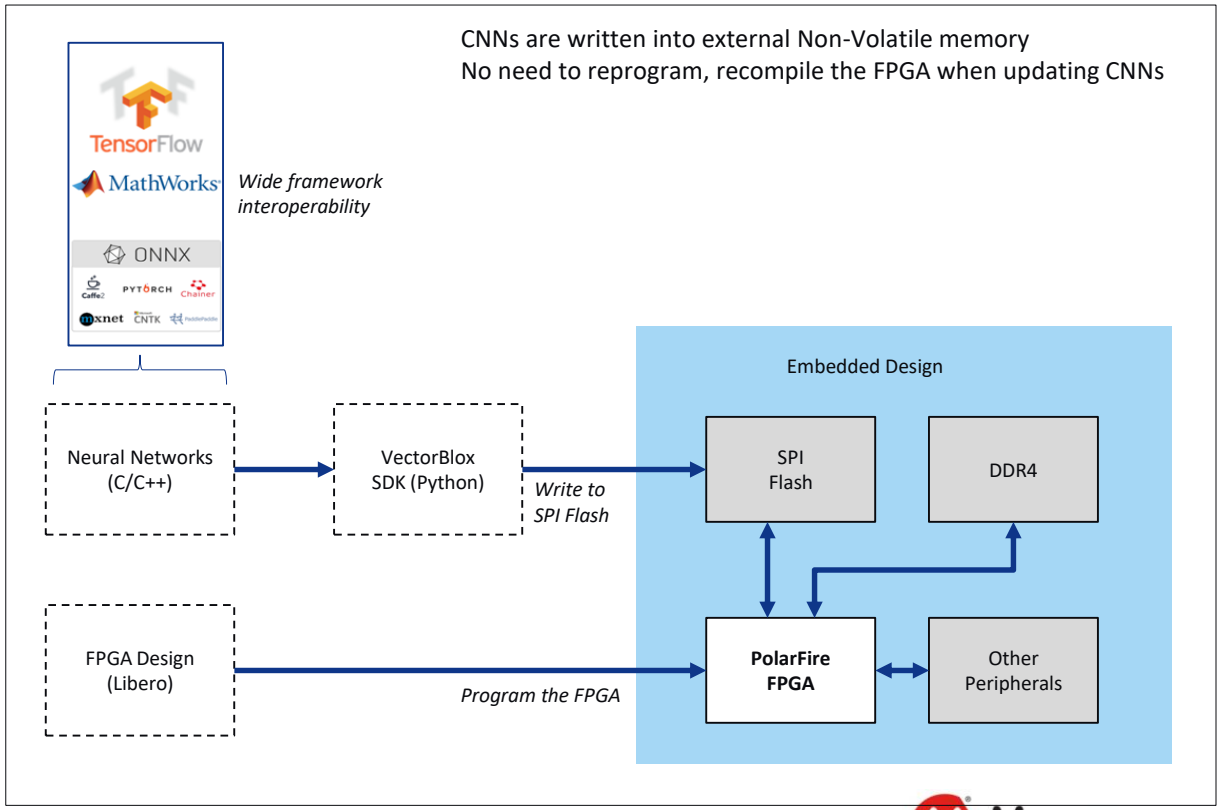
Bigger Brother: Machine Learning for Vision

VectorBlox™ SDK and NN IP enables

- Software developers to run Neural Networks (NN) without prior FPGA knowledge
- Utilization of most popular NN software frameworks
- Simulation in software without procuring hardware







TinyML

Implementation for Condition Monitoring

MICROCHIP

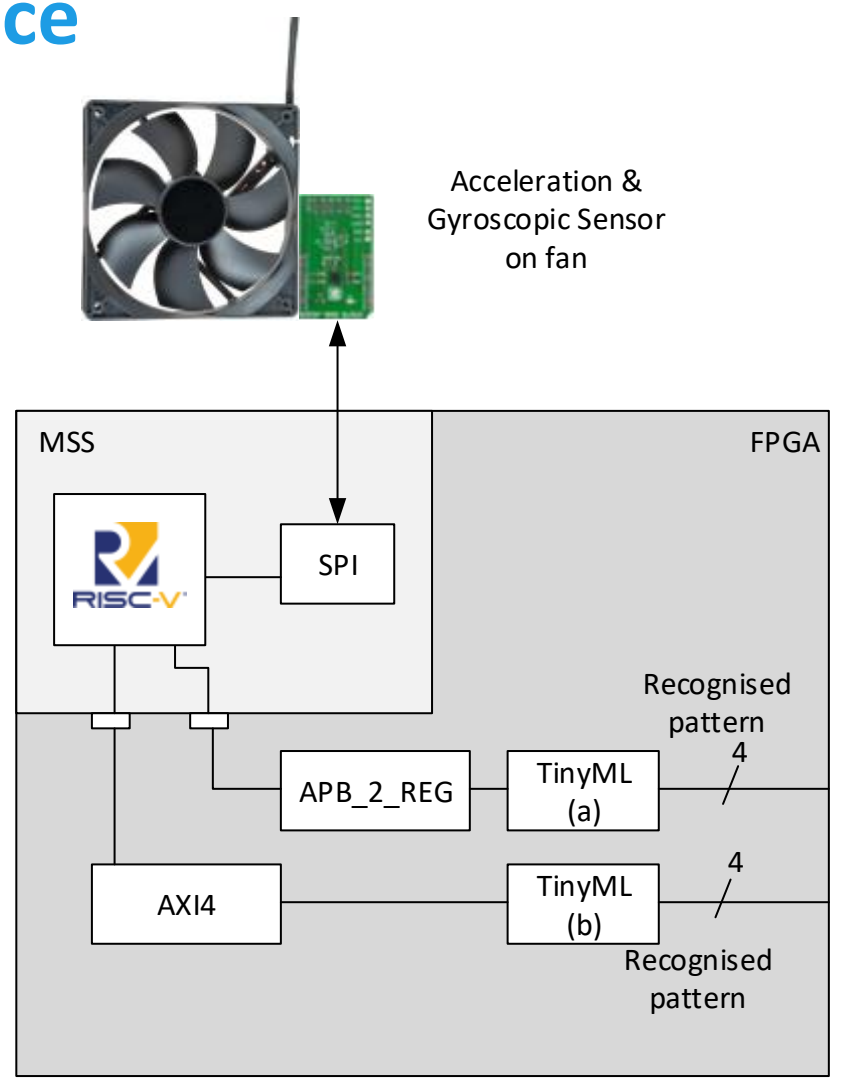
Goal for TinyML

- Small side-function as part of overall FPGA design
- Detection of anomalies on sensor data in system and classification into several „buckets“
- Work on simple 8-bit integer values from sensors
- TinyML was trained on INFXL—webpage based on recorded sensor data (cloud.infxl.com)
- Add additional value for existing fielded design as simple add-on with small logic size

Application Setup

Anomaly Detection for Predictive Maintenance

- **Bosch BMI160 sensor attached to fan**
- **Processor reading sensor via SPI and normalising to 8-bit data values**
- **TinyML trained on 8-bit sensor data**
- **Neural Network implemented using High Level Synthesis (HLS) tool LegUp (C-code to FPGA)**
- **Two variants for power comparison**
  - Simple FIFO interface
  - AXI4 based



Targetting FPGA Hardware

LegUp HLS Building the FPGA IP

- **Original Code Structure**

- **Adaption to FPGA, Register Interface**

- **Adaption to FPGA, AXI4 Interface**

```
int main() {
    int8_t ram8[ ROM16[0] + ROM16[2] ];
    while (1) {
        //get inputs, write them to ram8,
        //run forward pass, read outputs from ram8
        copy_inp_vec_to_ram8_head(ram8);
        deep_net_inference(       ram8);
        read_op_vec_frm_ram8_tail(ram8); }
    return 0;
}
```

```
void deep_net_inference(       int8_t ram8[]) {
    int8_t const static ROM8[] = { -126,-126,-126,-126,-126,-126,-126,-126,-126,-126,-126,-126,-126,-126,-126,-126,-126,-
    uint16_t            nr_ptr = ROM16[0] - 1;        //-1 nullifies the first pre-increment
    uint16_t         ROM16_ptr = 1;
    int16_t      static n_nr16, n_wt16, nr16, wt16, count16;
    for (    n_nr16 = ROM16[++ROM16_ptr], nr16 = 0; nr16 < n_nr16; ++nr16) { count16  =      ROM16[++ROM16_ptr];
        for (n_wt16 = ROM16[++ROM16_ptr], wt16 = 0; wt16 < n_wt16; ++wt16) { count16 += ram8[ROM16[++ROM16_ptr]]; }
        for (n_wt16 = ROM16[++ROM16_ptr], wt16 = 0; wt16 < n_wt16; ++wt16) { count16 -= ram8[ROM16[++ROM16_ptr]]; }
        if (     count16 <   0) { ram8[++nr_ptr] = -127; }
        else if (count16 > 656) { ram8[++nr_ptr] =  127; }
        else {                    ram8[++nr_ptr] = ROM8[count16];
        }
    }
}
```

```
int main() {
    FIFO<int8_t> batch;
    FIFO<int8_t> sensor;

    int8_t batch_display;
    for(int i=0;i<10;i++) {
        //get inputs, write them to ram8,
        //run forward pass, read outputs from ram8
        copy_inp_vec_to_ram8_head_mk_fifo(sensor);
        net_and_interface(sensor, batch);
        batch_display=batch.read();

        printf("\nResult: %x, %d", batch_display, batch_display);
    }
    return 0;
}
```

```
void net_and_interface(FIFO<int8_t> &sensor, FIFO<int8_t> &batch){
        #pragma LEGUP function top
        int8_t result[4];
        int8_t ram8[ROM16[0] + ROM16[2]];

        load_net(sensor, ram8);
        deep_net_inference2(ram8);
        read_op_vec_frm_ram8_tail_mk2(ram8, result);
        compare_value_fifo2(result, batch);

}
```

```
int main() {
    int8_t ram8[ ROM16[0] + ROM16[2] ];
    FIFO<int8_t> batch;
    FIFO<int8_t> sensor;

    int8_t batch_display;
    for(int i=0;i<10;i++) {
        //get inputs, write them to ram8,
        //run forward pass, read outputs from ram8
        copy_inp_vec_to_ram8_head_mk(ram8);
        net_and_interface_axi4(batch);
        batch_display=batch.read();

        printf("\nResult: %x, %d", batch_display, batch_display);
    }
    return 0;
}
```

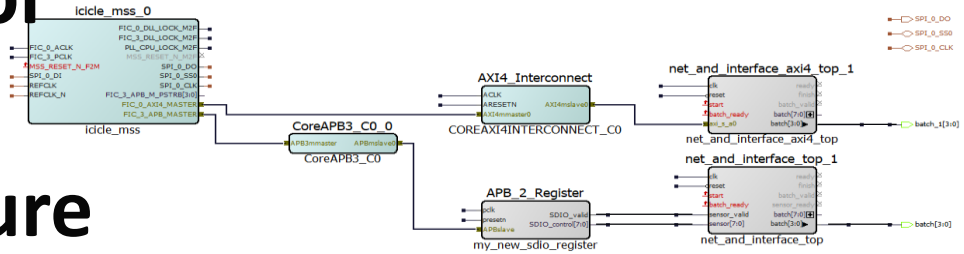
```
void net_and_interface_axi4(FIFO<int8_t> &batch){
        #pragma LEGUP function top
        int8_t result[4];
        int8_t ram8[ROM16[0] + ROM16[2]];

        load_net_axi4(ram8);
        deep_net_inference(ram8);
        read_op_vec_frm_ram8_tail_mk2(ram8, result);
        compare_value_fifo(result, batch);

}
```


MICROCHIP

Targetting PolarFire® SoC

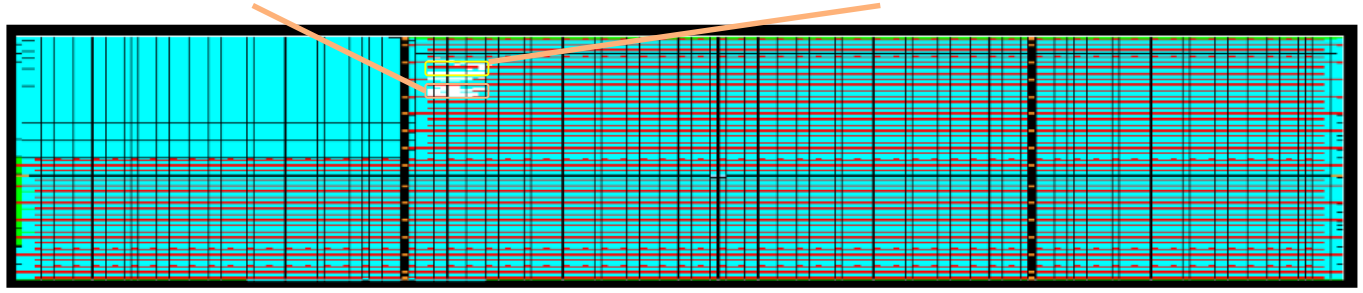
Icicle Kit as Target Board

- **Board used because of mikroBus™ socket**
- **TinyML via AXI4 and register interface for size comparison**
- **Implementation size include infrastructure for interfacing with processor**



IP-Name	Logic	FF	LSRAM
AXI4 Interconnect	408	666	0
TinyML + AXI4	636	916	1

IP-Name	Logic	FF	LSRAM
APB 2 Register	5	18	0
TinyML (Register)	353	420	1
(Neural Net alone)	(240)	(340)	(1)



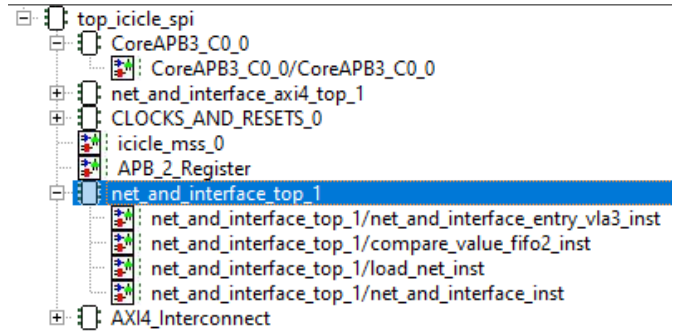
Post implementation layout in PolarFire SoC MPFS250T

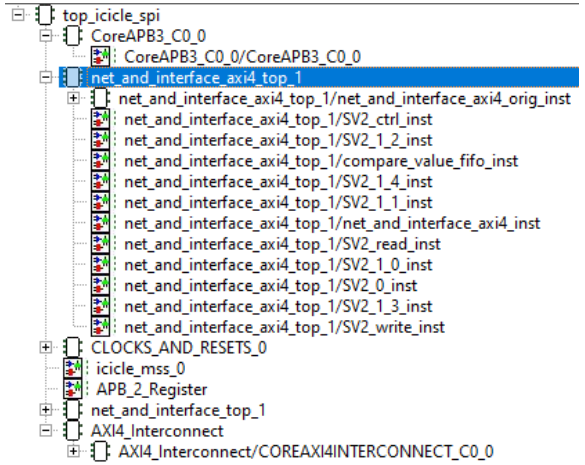
MICROCHIP

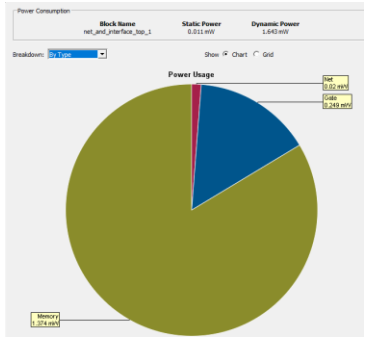
Power Analysis TinyML

Tiny ML and its Sub-Blocks

- **Power analysis done with SmartPower (part of Libero® SoC implementation software)**

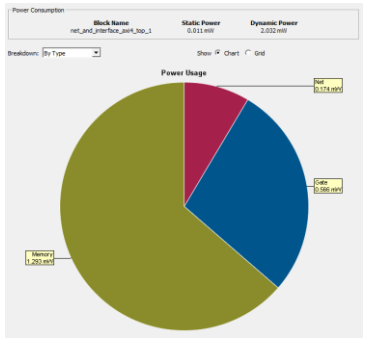






Power Consumption
- LSRAM: 1.37 mW
- Logic: 0.25 mW

Activities based on vectorless analysis



Power Consumption (incl AXI4 Interconnect)
- LSRAM: 1.29 mW
- Logic: 1.05 mW

Performance Estimation

- **Hardware / Software Co-Simulation Based on LegUp HLS**
  - One recognition cycle approximately 350 FPGA clock-cycles
  - At 100 MHz = 3.5 µs
- **Power Estimation for One Recognition (ML alone):**
  - 3.5µs * 1.62 mW = 5.67 *10^{-9} Ws = 5.67 nJ

Summary

- TinyML from INFXL simple to implement in Microchip low-power FPGAs using LegUp HLS
- Very small footprint and low power consumption
- Allow condition monitoring to be a small side-function inside the main FPGA functionality

Questions?

Martin.Kellermann@microchip.com
https://www.linkedin.com/in/martinkellermann

Thank You

Copyright Notice

www.tinyML.org