TinyML

Why ML on tiny devices is a major thing

Morten Opprud Jakobsen, PhD student @AU ECE, morten@ece.au.dk

Også i jylland...

TINY ML

meetup COPENHAGEN

Part of **tinyML – 34 groups**

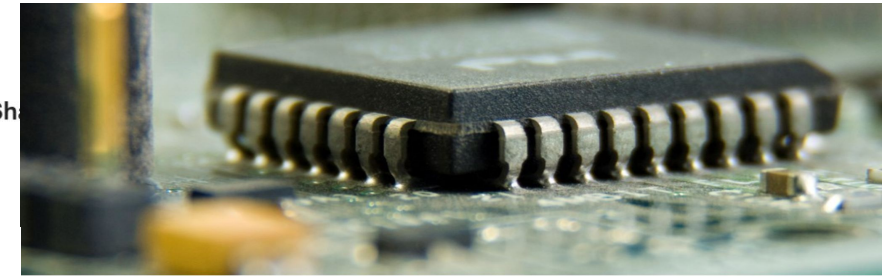
tinyML: Enabling ultra-low Power ML at the edge - Copenhagen

Copenhagen, Denmark

131 members · Public group

Organized by **Olga G.** and **3 others**

Sh



IDA

IDA Embedded

IDA Embedded er et socialt og fagligt netværk, der følger og aktivt deltager i udviklingen på det datatekniske fagområde.

Pris

IDA-medlem	Gratis
Ikke IDA-medlem	500 kr. pr. år

Tilmeld

What is embedded machine learning / TinyML?

- Running machine learning algorithms on embedded systems
- Why do we need this?
  - Classification, prediction, decision making with little or no Internet connection
  - Voice activation, object detect, anomaly detection, etc.
  - Anonymising / compression of raw data

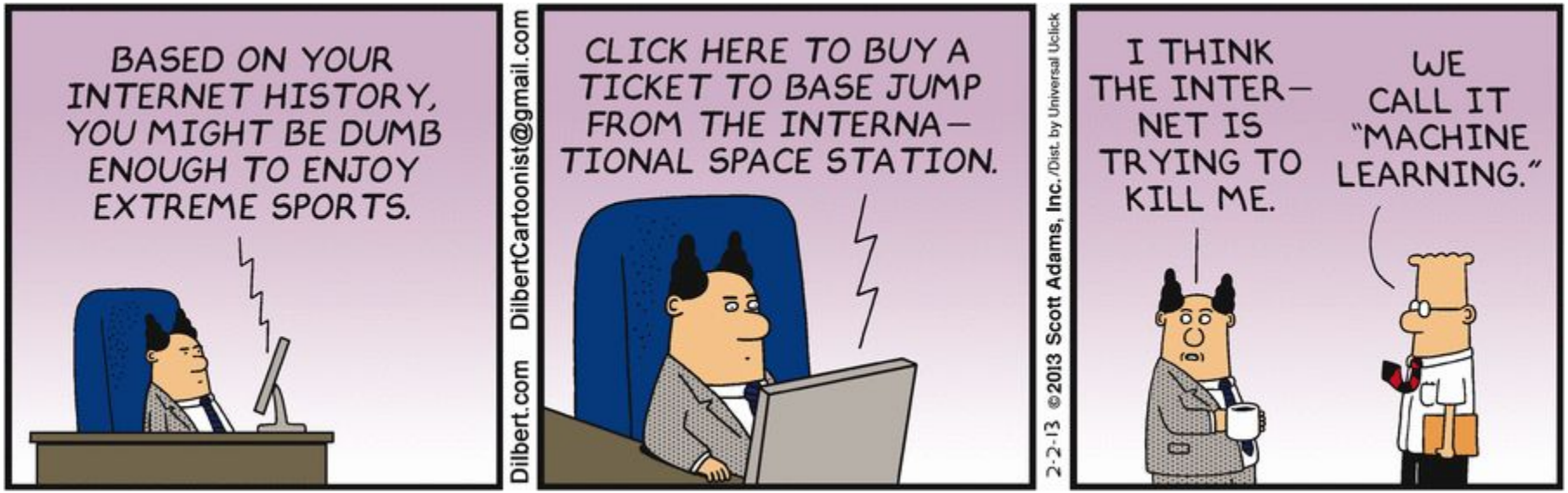
https://docs.google.com/presentation/d/1E_Wod_Ag1JZIwDxhq3-ZSlooA_CxajjtcRWZzuzBdWQ/edit#slide=id.ga0a9a3bbef_0_425

What is TinyML

- **MachineLearning**
-- ML vs AI
-- Learning vs. inference
-- ML vs Sequential programming
-- 5 min hands on
- TinyML
-- Motivation
-- Example use cases
-- Technological enablers
-- Hardwaregoogle slides inert page

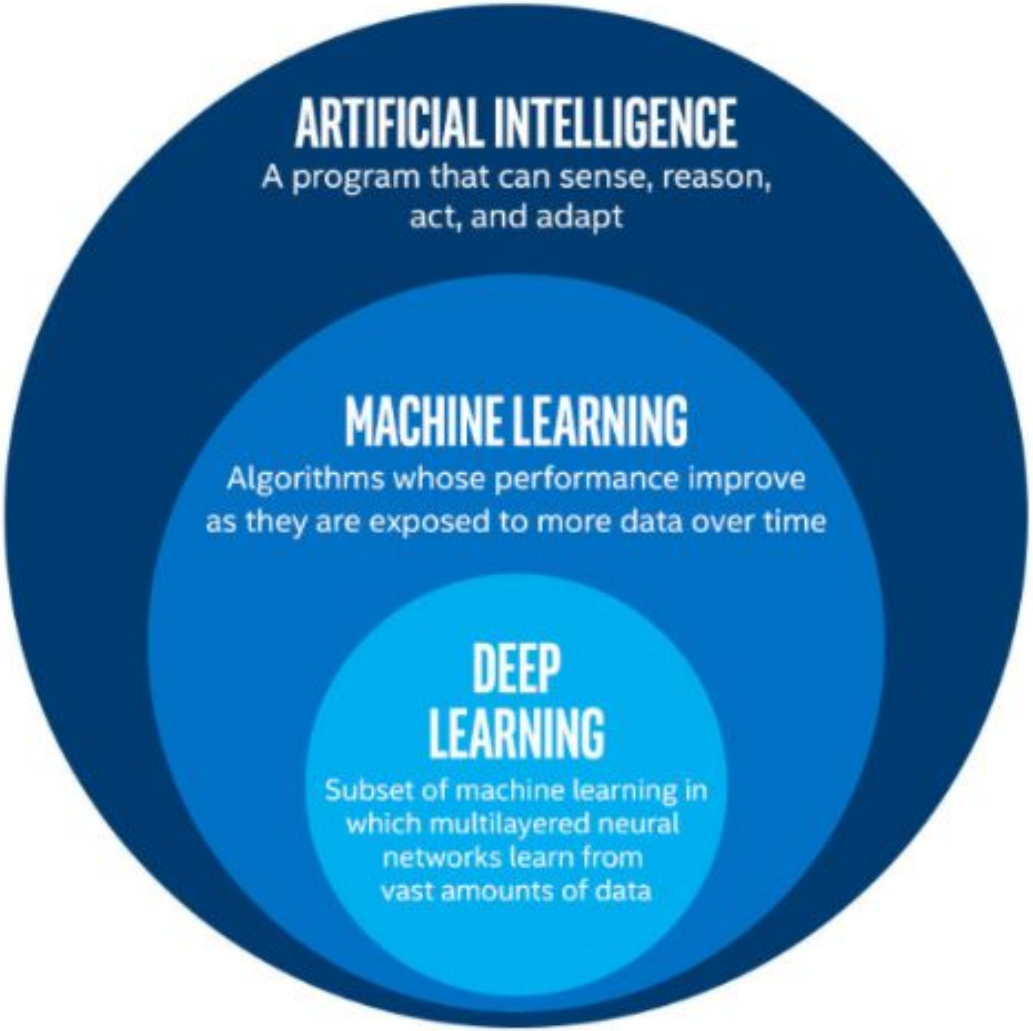
-- Demo / example
-- Models - big to small
-- Software
-- Tools
-- Quiz
- Resources
- Join
- Q&A

slides:https://github.com/opprud/tinyML/



machine learning

Artifici *hich we can*
create *human*
intellig



Machi *elligence,*
which *data or*
exper *nmed.*

Learning vs. Inference

Machine learning occurs in two stages – learning (training) and inferencing.

At present, TinyML only handles inferencing.

Learn / Train

During learning, the ML model adjusts its internal configurations based on the data that it receives in order to achieve a better result on its given task.

To achieve this, the data is passed forwards through the model, where a loss is calculated. The feedback from this loss is then passed backwards through the model for the adjustments to be made.

https://prince-canuma.medium.com/understanding-neural-networks-22b29755abd9

This is repeated for multiple inputs, up to billions or even trillions of times!

As you can imagine, this is extremely resource intensive and difficult to perform effectively even on some laptops, let alone microcontrollers with drastic limitations in computing resources.

Quiz

What's the compute performance (FLoating Point Ops per sec - FLOP's) of a single google v2 Tensor Processing Unit - TPU ?

45 Tera Flops (1 Tera = 10^12)

https://en.wikipedia.org/wiki/Tensor_Processing_Unit#Fourth_generation_TPU

Inference

Inference, on the other hand, refers to using the model to make some conclusions on input data.

For example, we might provide a weather predictor with some temperature or humidity values to receive a prediction for whether it will rain.

While state of the art machine learning models still require a significant amount of hardware to run effectively, it's possible to optimise models for lightweight inferencing on the edge

– this is precisely where TinyML comes in!

https://www.seeedstudio.com/blog/2021/06/14/everything-about-tinyml-basics-courses-projects-more/

Quiz

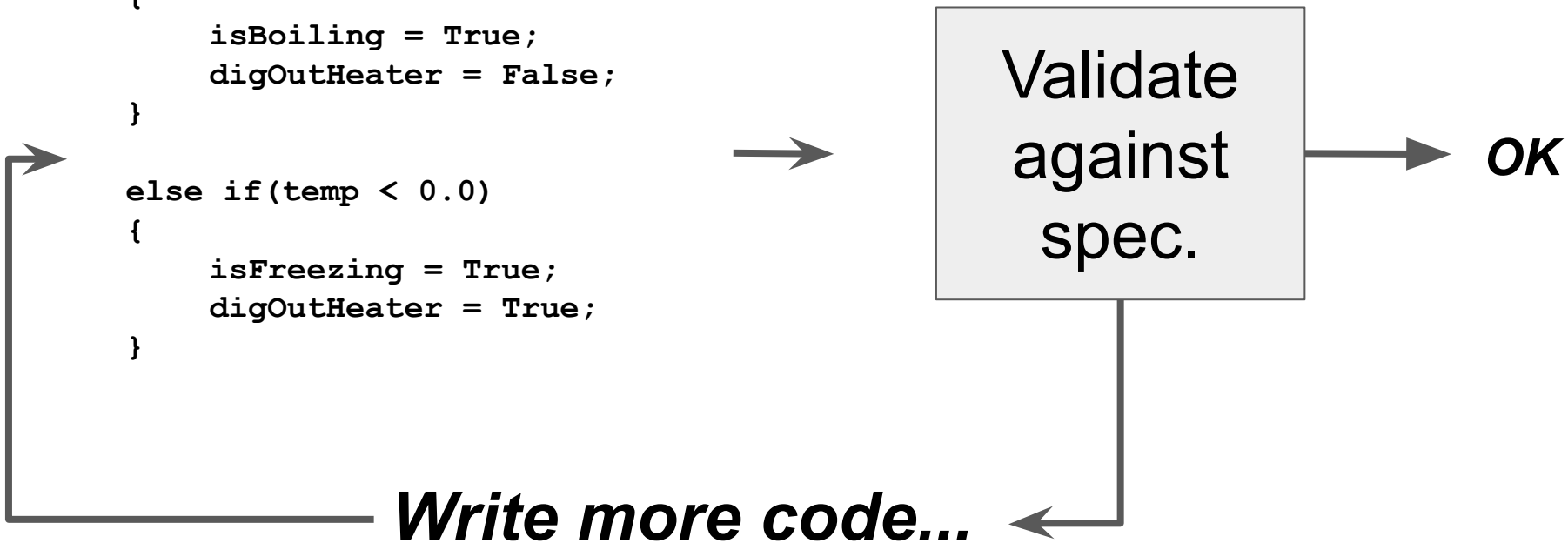
What's the compute performance (FLoating Point Ops per sec - FLOP's) of a single google Edge TPU ?

4 Tera Flops (@2W)

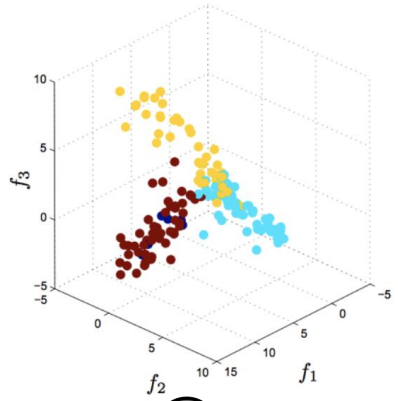
https://coral.ai/technology/

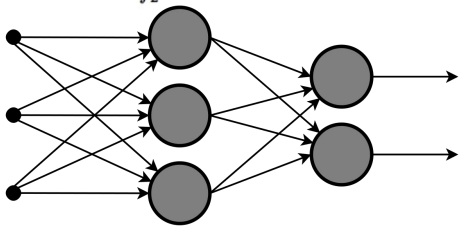
Rule based (deterministic) programming

```
if(temp > 100.0)
{
     isBoiling = True;
     digOutHeater = False;
}

else if(temp < 0.0)
{
     isFreezing = True;
     digOutHeater = True;
}
```

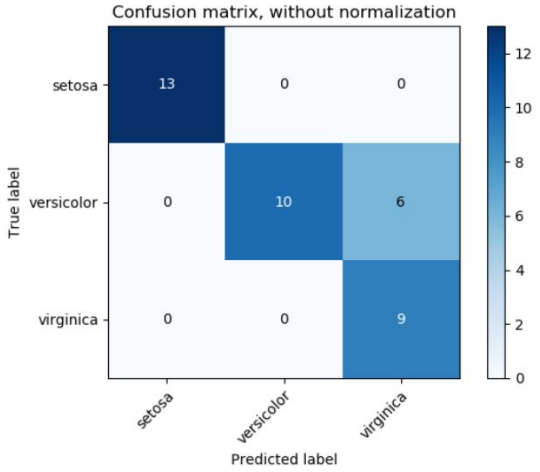


Data driven programming / ML





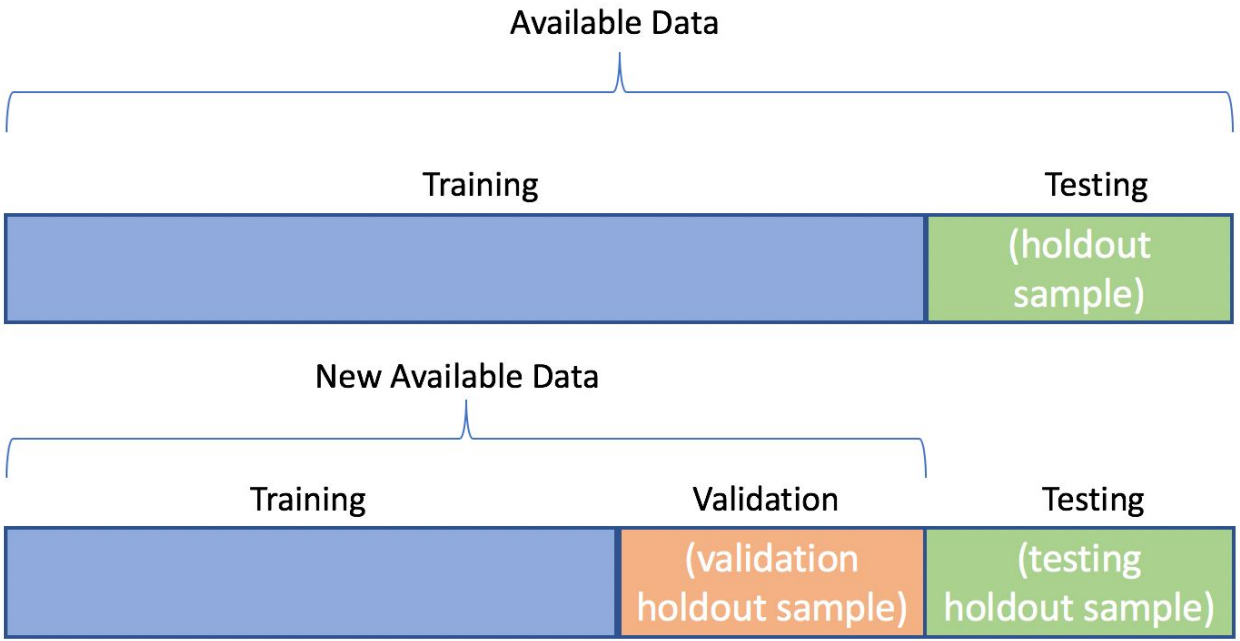
```
res = do_ML_Inference(data

switch(res):

case ´1´:
     //do something
case ´2´:
     //do something else
```



OK

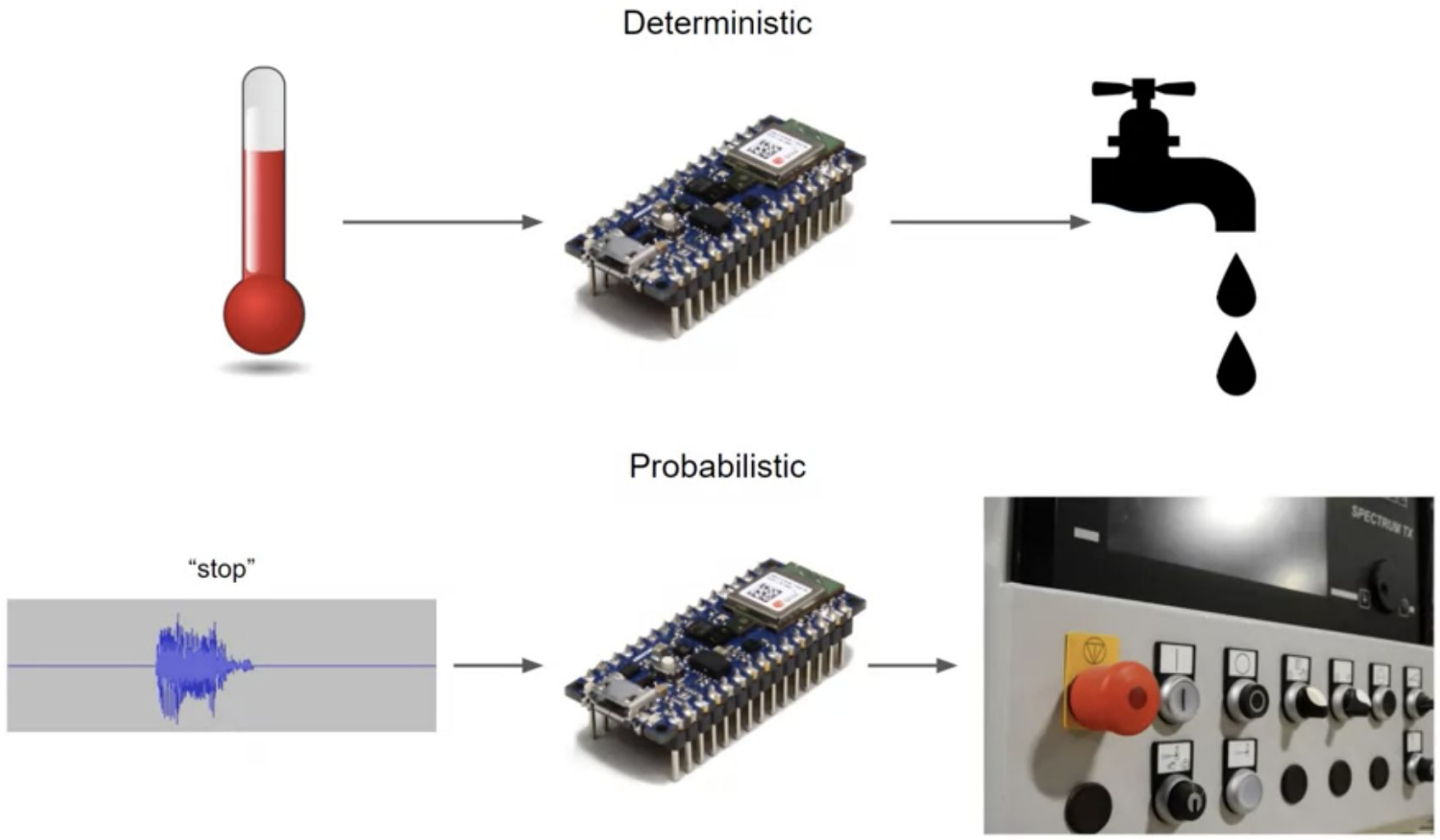
Collect more data...

Split data for Test / validation



Models can "overfit" on known data, therefore we put some data aside the model has never seen before for testing

U said ‘up’ or ‘stop’?



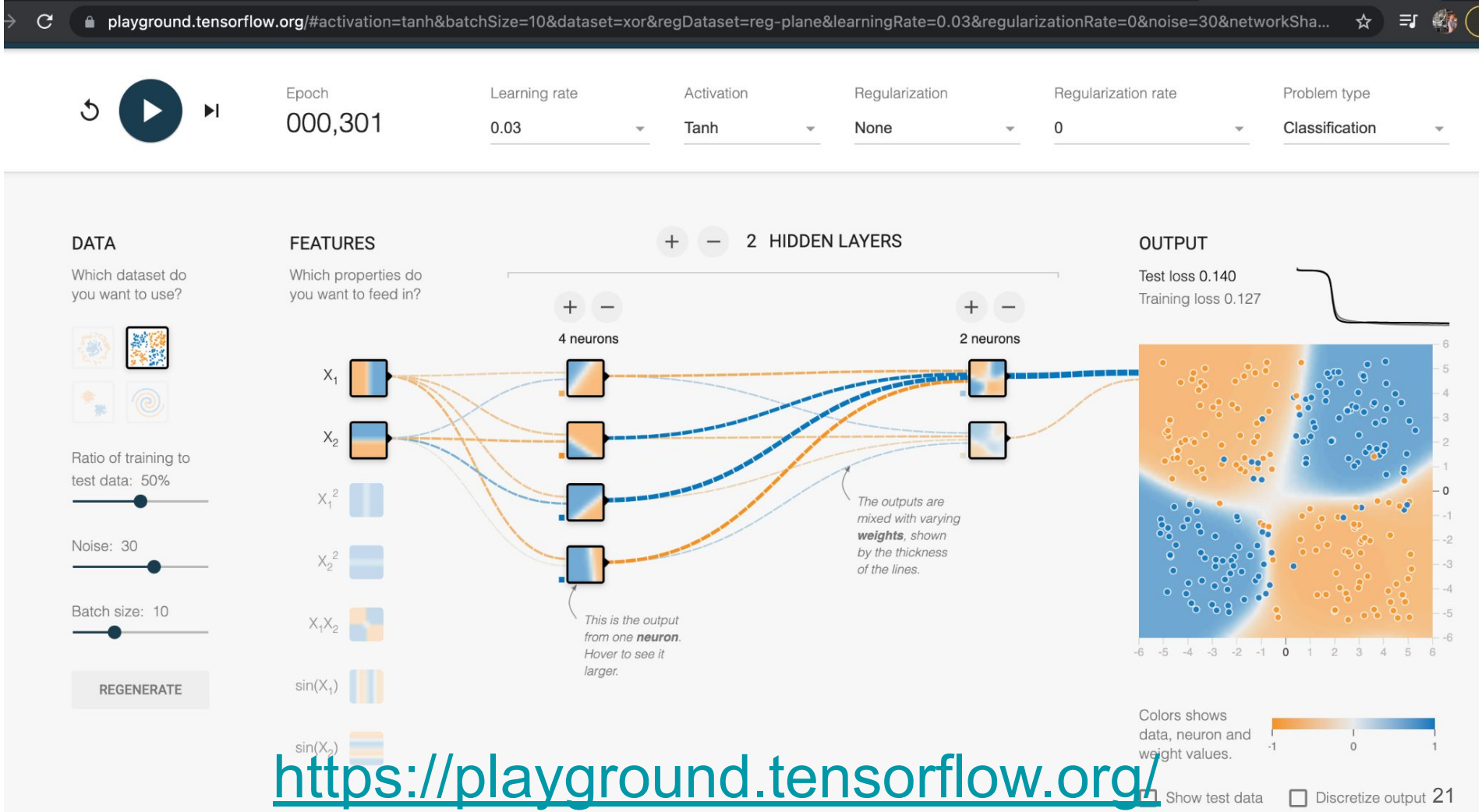
Rule based / traditional programming

Data driven / ML programming

"– *ML is just curve fitting"*

Spend 5 min at TF playground at:
https://playground.tensorflow.org/

More at: https://cs.stanford.edu/people/karpathy/convnetjs/


playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=xor&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=30&networkSha...
Epoch
000,301
Learning rate
0.03
Activation
Tanh
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 30
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
X1
X2
X1²
X2²
X1X2
sin(X1)
sin(X2)
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
OUTPUT
Test loss 0.140
Training loss 0.127
Colors shows data, neuron and weight values.
-1
0
1
Show test data
Discretize output
https://playground.tensorflow.org/
21

What is TinyML

- MachineLearning
-- ML vs AI
-- Learning vs. inference
-- ML vs Sequential programming
-- 5 min hands on
- **TinyML**
-- Motivation
-- Example use cases
-- Technological enablers
-- Hardware
-- Demo / example
-- Models - big to small
-- Software
-- Tools
-- Quiz
- Resources
- Join
- Q&A

slides:https://github.com/opprud/tinyML/

About TinyML

TinyML is one of the fastest-growing areas of Deep Learning. In a nutshell, it's an emerging field of study that explores the types of models you can run on small, low-power devices like microcontrollers.

TinyML sits at the intersection of embedded-ML applications, **algorithms**, **hardware** and **software**.

The goal is to enable low-latency inference at edge devices on devices that typically consume only a few **milliwatts** of battery power

Motivation - Low power designs

“The physics of moving data around just seems to require a lot of energy. There seems to be a rule that the energy an operation takes is proportional to how far you have to send the bits”

https://petewarden.com/2018/06/11/why-the-future-of-machine-learning-is-tiny/

Motivation - Low bandwidth

Since data does not constantly have to be sent to a server, less wireless bandwidth is used. By sending only results LPWAN technologies like Zigbee, NBIoT and LoRa can be considered

Wireless is expensive

ESP32 Series

Datasheet

Including:

- ESP32-D0WD-V3
- ESP32-D0WDQ6-V3
- ESP32-D0WD
- ESP32-D0WDQ6
- ESP32-D2WD
- ESP32-S0WD
- ESP32-U4WDH

5.5 RF Power-Consumption Specifications

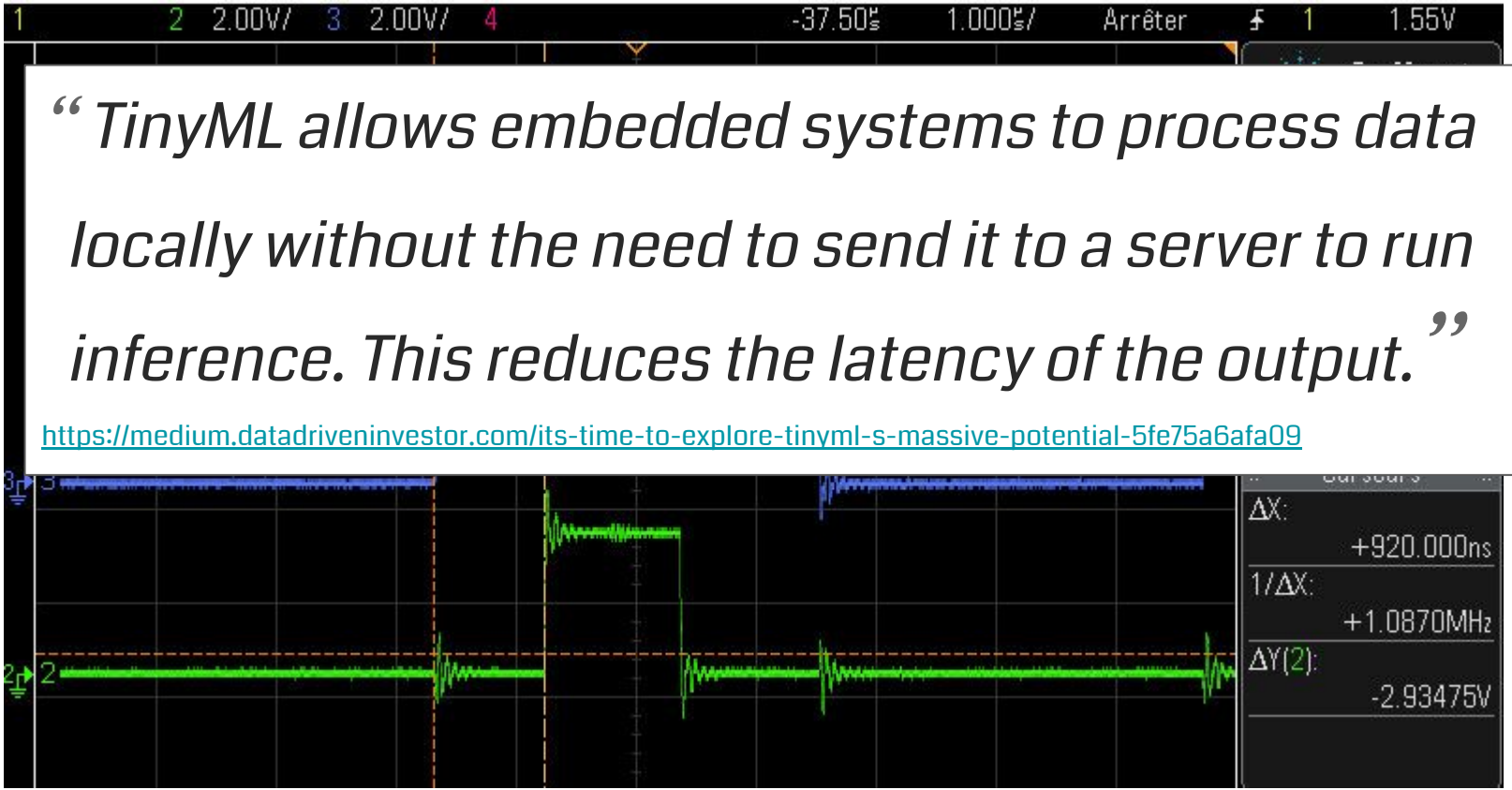
300-800mW

The power consumption measurements are taken with a 3.3 V supply at 25 °C of ambient temperature at the RF port. All transmitters' measurements are based on a 50% duty cycle.

Table 15: RF Power-Consumption Specifications

Mode	Min	Typ	Max	Unit
Transmit 802.11b, DSSS 1 Mbps, POUT = +19.5 dBm	-	240	-	mA
Transmit 802.11g, OFDM 54 Mbps, POUT = +16 dBm	-	190	-	mA
Transmit 802.11n, OFDM MCS7, POUT = +14 dBm	-	180	-	mA
Receive 802.11b/g/n	-	95 ~ 100	-	mA
Transmit BT/BLE, POUT = 0 dBm	-	130	-	mA
Receive BT/BLE	-	95 ~ 100	-	mA

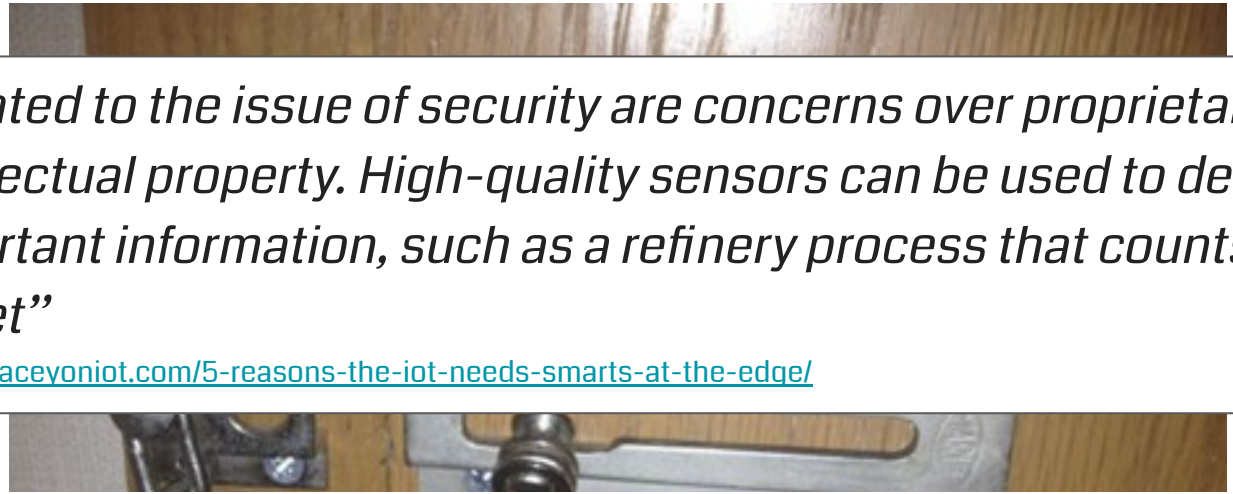
Motivation - Low latency / resilience



“ TinyML allows embedded systems to process data locally without the need to send it to a server to run inference. This reduces the latency of the output.”

https://medium.datadriveninvestor.com/its-time-to-explore-tinyml-s-massive-potential-5fe75a6afa09

Motivation - Privacy / Security



“Related to the issue of security are concerns over proprietary data and intellectual property. High-quality sensors can be used to derive important information, such as a refinery process that counts as a trade secret”

https://staceyoniot.com/5-reasons-the-iot-needs-smarts-at-the-edge/

“An example of using custom wake words to activate devices or new interfaces that could include gestures or gaze detection. By using on-device ML, such interfaces would also preserve user privacy”

https://staceyoniot.com/privacy-and-new-functions-will-make-tinyml-big/

README.m

Magic

Magic Wanc

Demo V

Click the im

Prerequ

- Platforr
- platforr

using

TinyML Example: Anomaly Detection

This project is an example demonstrating how to use Python to train two different machine learning models to detect anomalies in an electric motor. The first model relies on the classic machine learning technique of Mahalanobis distance. The second model is an autoencoder neural network created with TensorFlow and Keras.

Data was captured using an ESP32 and MSA301 3-axis accelerometer taped to a ceiling fan. Each sample is about 200 samples of all 3 axes captured over the course of 1 second. Fan was run at multiple speeds (off, low, medium, high) with and without a weight. 1 "weight" is one US quarter taped to one of the fan's blades to create an offset motion. All raw data is stored in the ceiling-fan-dataset directory.

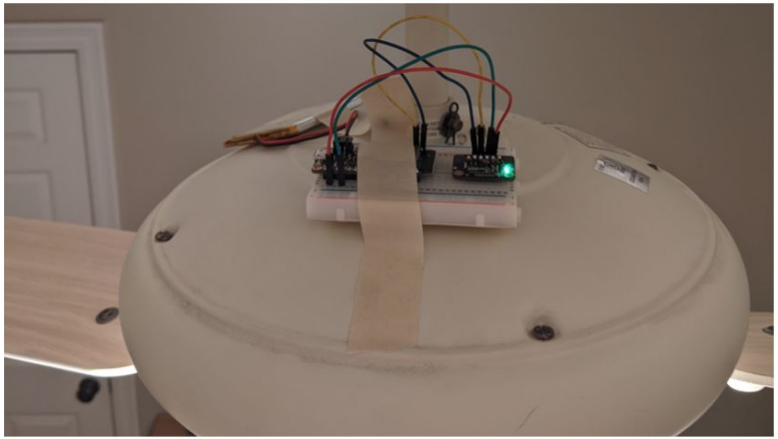
Note that if you create a "robust" model by moving the fan around during data collection, the Autoencoder works much better than the Mahalanobis Distance method.

The full articles that explain how these programs work and how to use them can be found here:

- Edge AI Anomaly Detection Part 1 - Data Collection
- Edge AI Anomaly Detection Part 2 - Feature Extraction and Model Training
- Edge AI Anomaly Detection Part 3 - Machine Learning on Raspberry Pi
- Edge AI Anomaly Detection Part 4 - Machine Learning Models on Arduino

Here are the accompanying YouTube videos that explain how to use these programs and some of the theory behind them:

- Edge AI Anomaly Detection Part 1: Data Collection
- Edge AI Anomaly Detection Part 2: Feature Extraction and Model Training
- Edge AI Anomaly Detection Part 3: Deploy Machine Learning Models to Raspberry Pi
- Edge AI Anomaly Detection Part 4: Deploy TinyML Model in Arduino to ESP32



latformIO.

ote that I'm

Technology “enablers” for TinyML applications

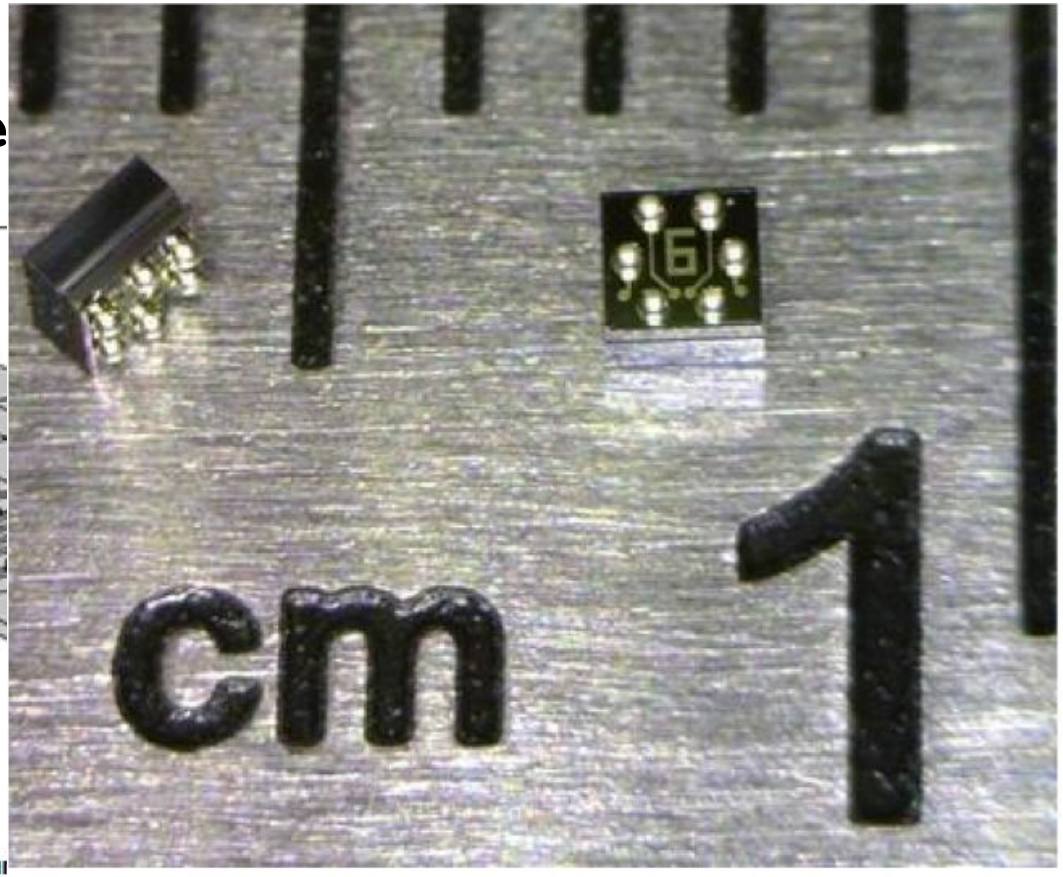
1 -“Sense” advanced, and cheap MEMS sensors

2 -“Compute” cheap and efficient processing

3 -“Think” Signal processing & Machine Learning

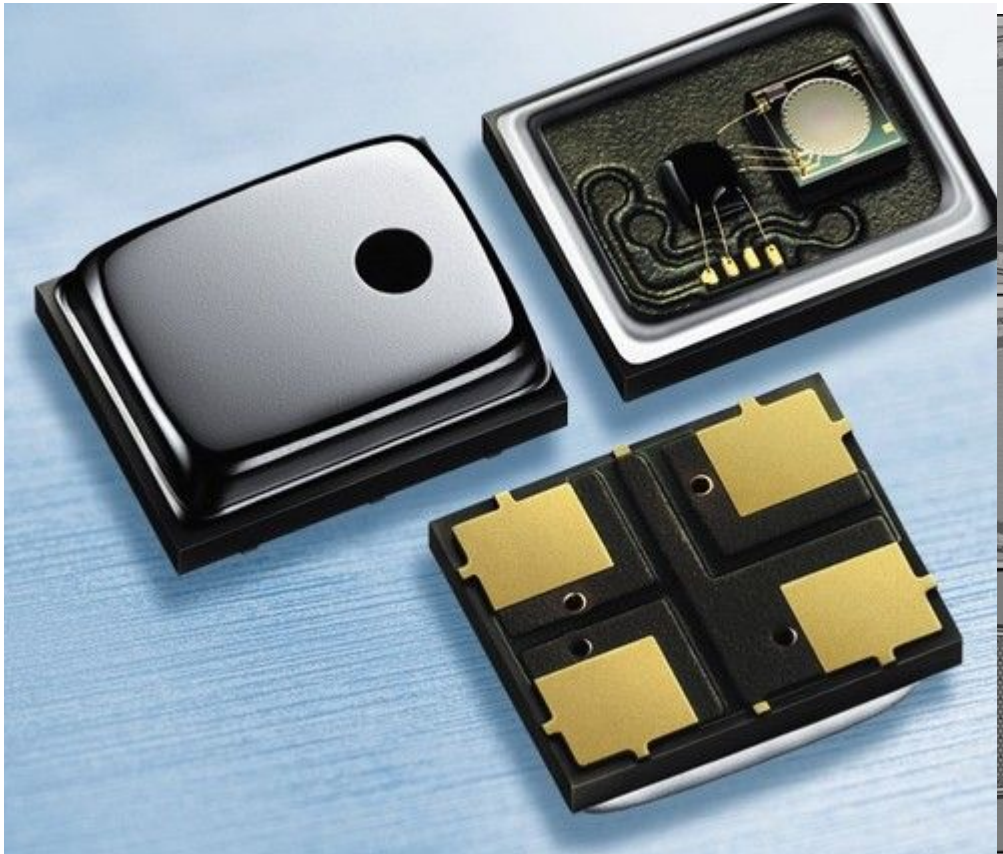
4- “Transmit” Wireless Long Range technologies

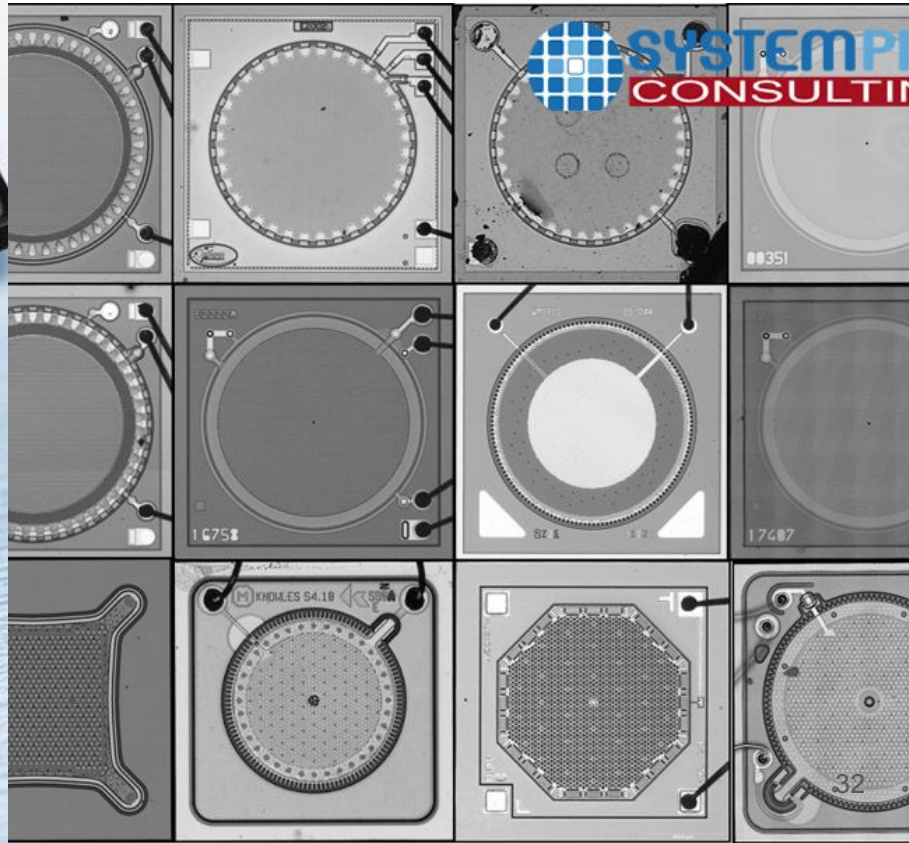
1 - Sense nsorer



mCube MC3571 1x1mm Accelerometer

1 - Sense - avancerede og billige sensorer



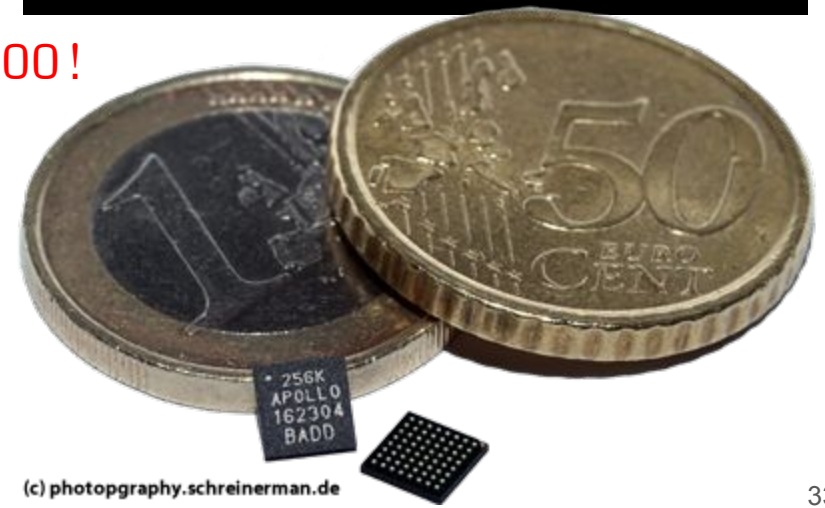


2 - Compute

Pentium P54C 166Mhz 1994
- FFT 1024 166 per second
- Cost $1,500
- 100's Watts

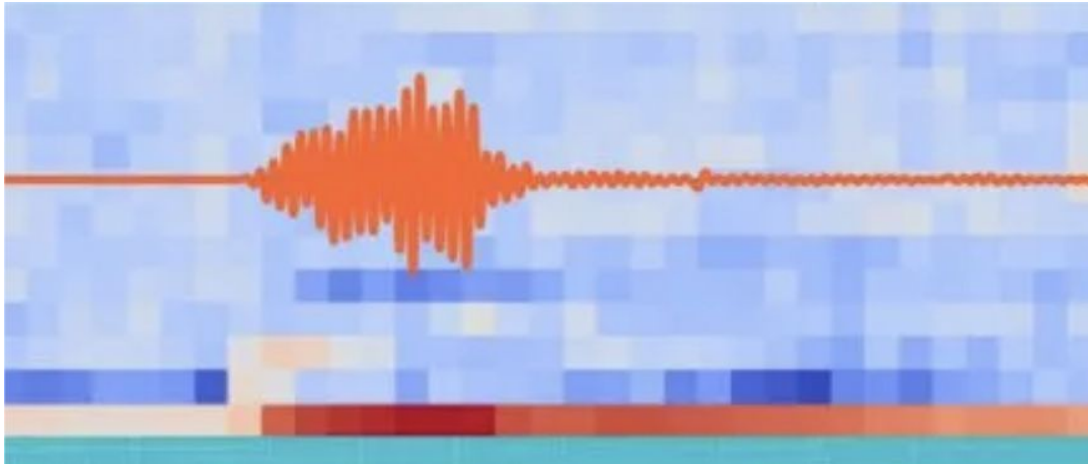
ARM Cortex M4 processor
- FFT 1024 800 per second
- Cost less that <$3
- sub-mWpower draw

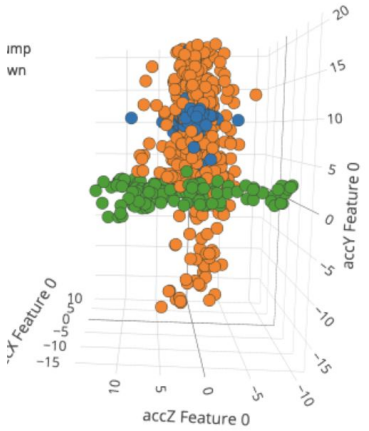
100 W / 1mW = 100.000 !

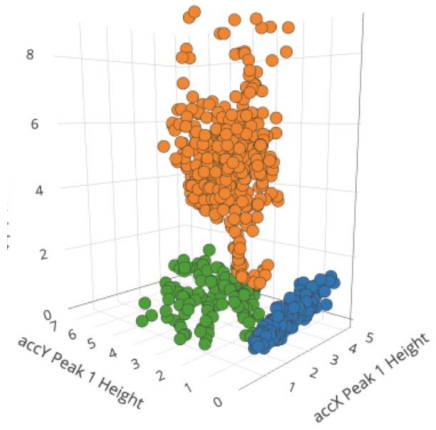


(c) photopgraphy.schreinerman.de

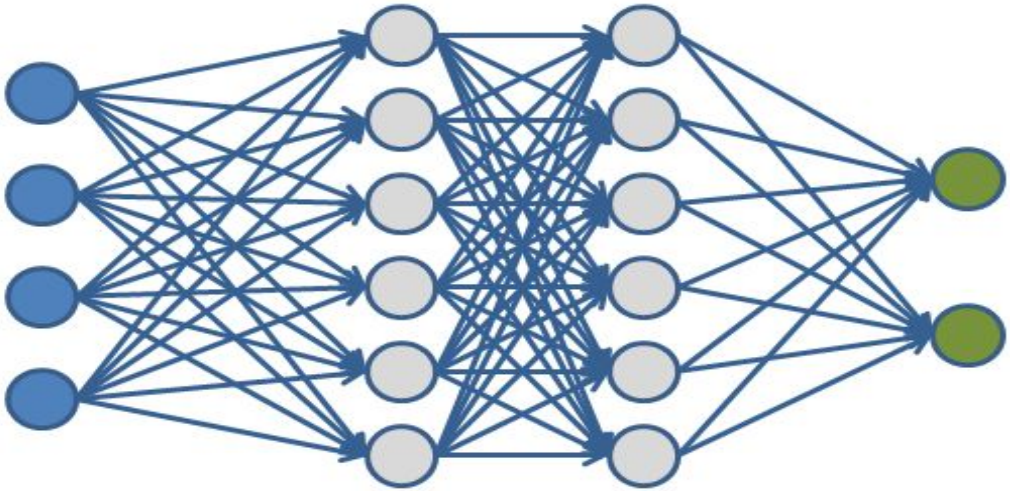
3 - “Think” signal processing

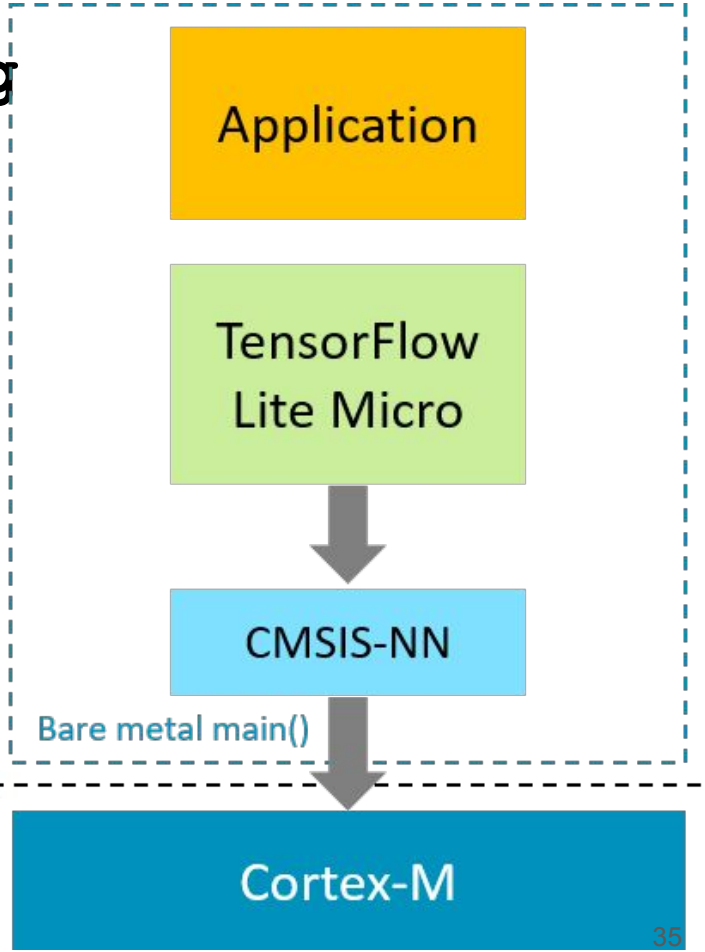




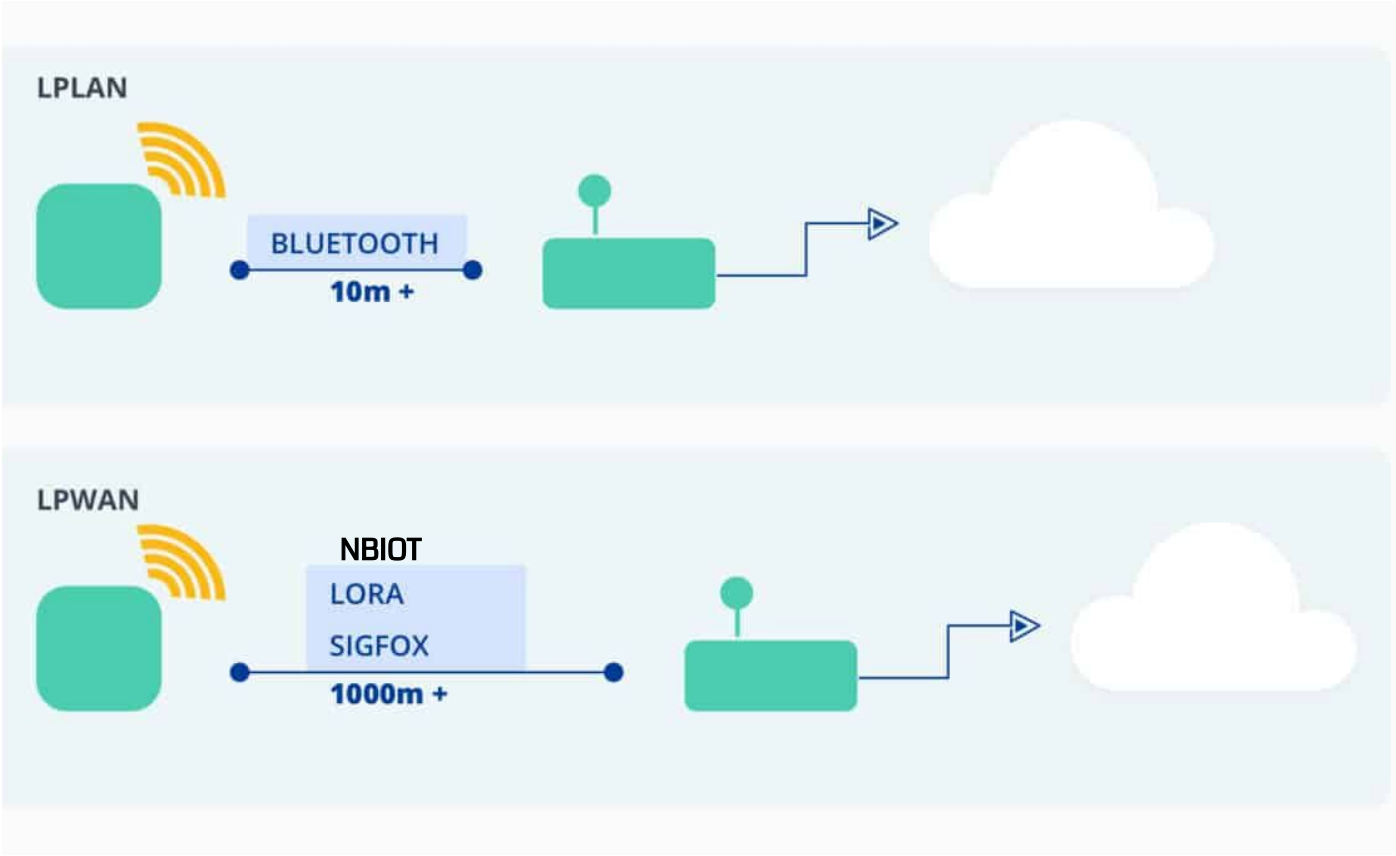


3 - "Think" Machine Learning

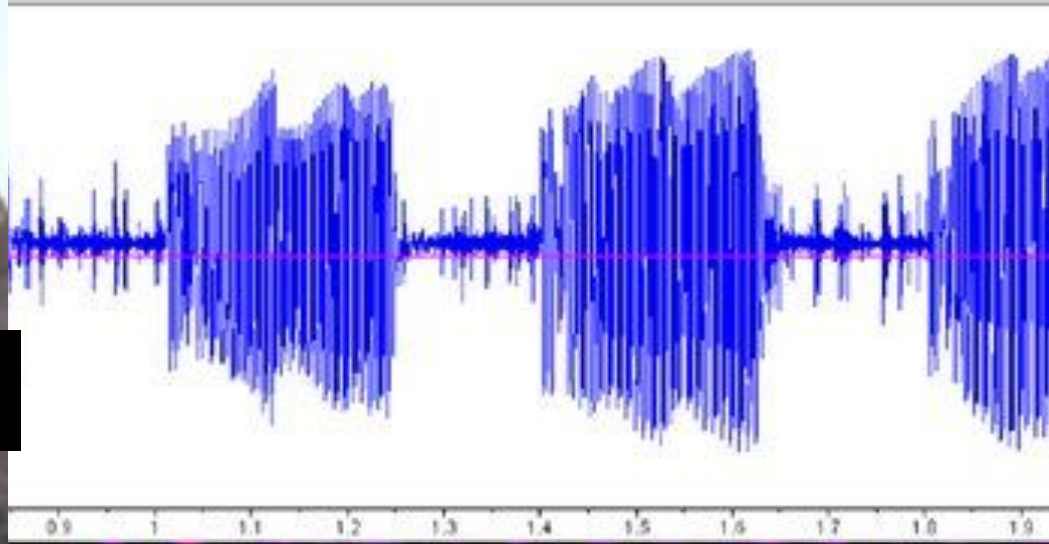


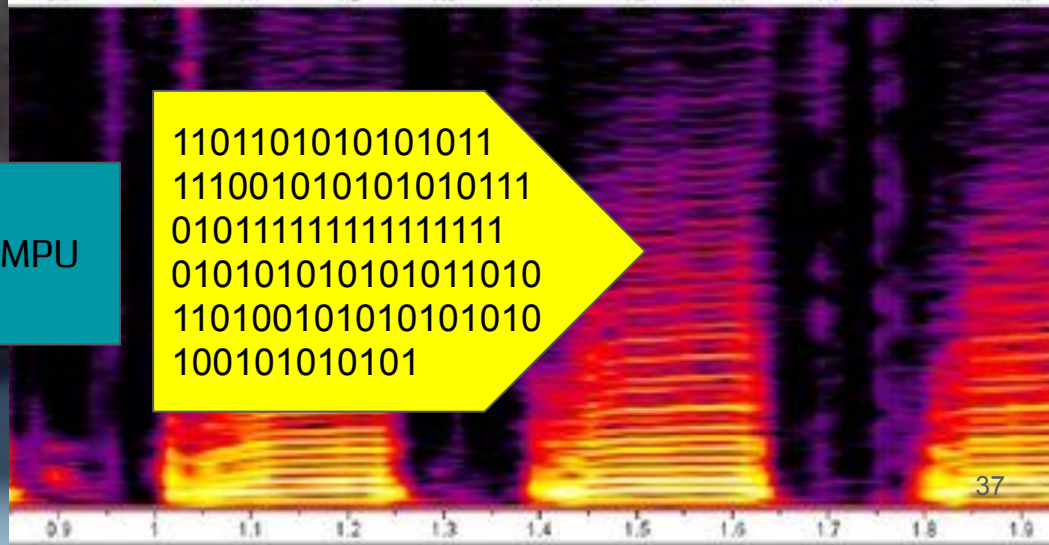


4 - "Transmit" LPWAN / LPLAN



“Un-intelligent sensor”

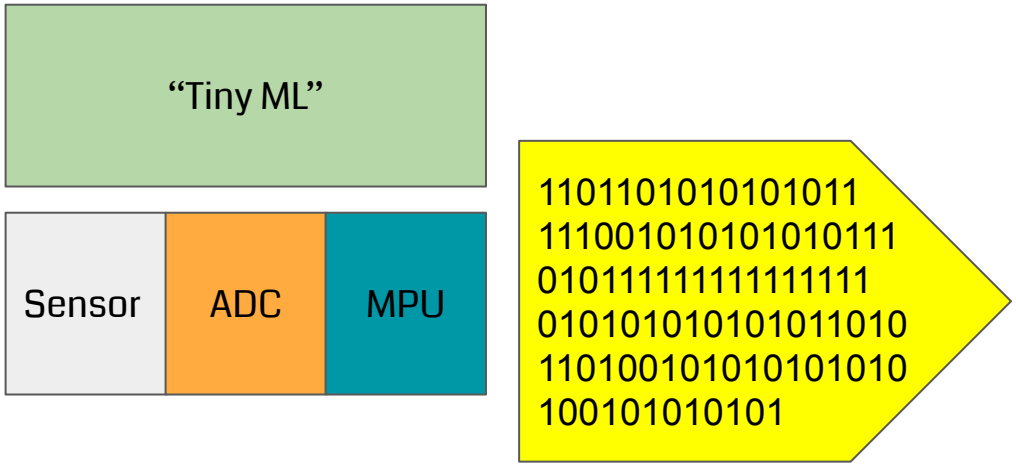




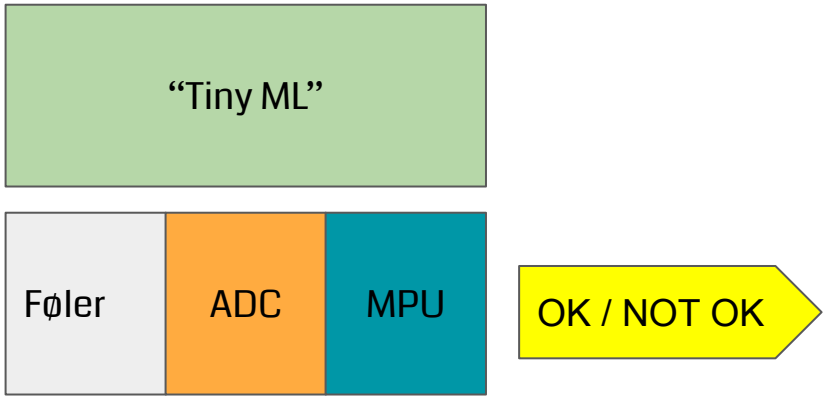
Føler → ADC → MPU

1101101010101011
1110010101010111
0101111111111111
0101010101011010
1101001010101010
100101010101

Sense + Compute + Think + Transmit



Sense + Compute + Think + Transmit



What is TinyML

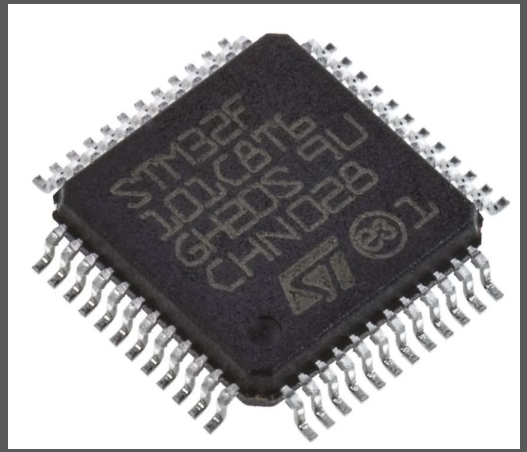
- MachineLearning
-- ML vs AI
-- Learning vs. inference
-- ML vs Sequential programming
-- 5 min hands on
- TinyML
-- Motivation
-- Example use cases
-- Technological enablers
-- **Hardware**
-- Demo / example
-- Models - big to small
-- Software
-- Tools
-- Quiz
- Resources
- Join
- Q&A

slides:https://github.com/opprud/tinyML/

Typical power consumption (Watt)		Typical HW	Train	Inference
100 - 1000's	ML on cloud backend	GPU/TPU cluster	✓	✓
10 - 100	ML on PC / laptop	4-16 core computer	✓	✓
1 - 10	Edge ML	SBC	(✓) ✓	
>1mW	Tiny ML	MCU Or custom ASIC	✗	✓

Quiz

What's the compute performance (FLoating Point Ops per sec - FLOP's) of a typical Cortex M4F CPU?

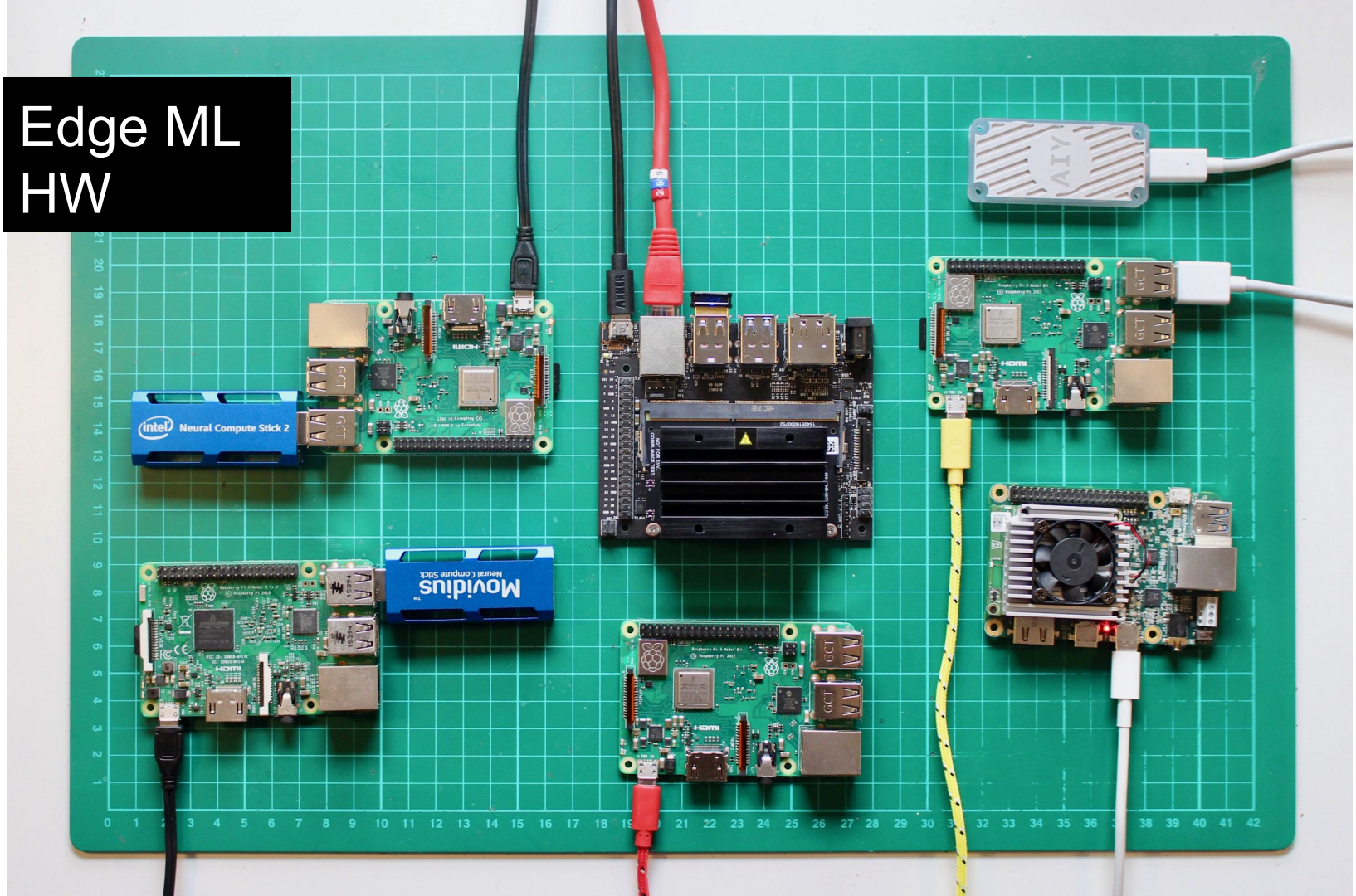


1-3 cycles/instruction (e.g 576uW@96Mhz)

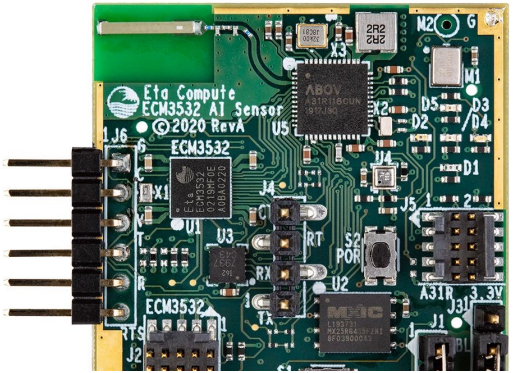
https://community.arm.com/developer/ip-products/processors/f/cortex-m-forum/9722/mflops-of-m4f/

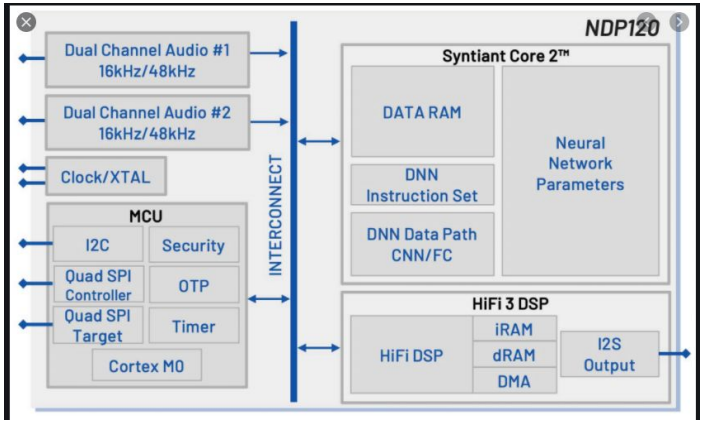
Compute power	Typical power consumption (Watt)		Typical HW	Train	Inference
50TFLOPS - 100PFLOPS	100 - 1000	ML on cloud backend	GPU cluster	✓	✓
	10 - 100	ML on PC / laptop	4-16 core computer	✓	✓
4TFLOPS	1 - 10	Edge ML	SBC	(✓) ✓	
1/10.000 TFLOPS	>1mW	Tiny ML	MCU Or custom ASIC	✗	✓

Edge ML HW



TinyML HW - ultra low power





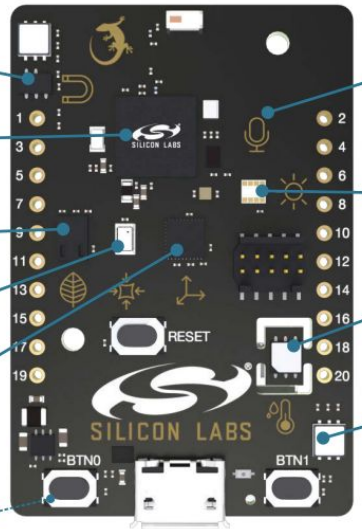
Magnetic Field Sensing
Hall effect sensor
Silicon Labs Si7210

Communication and Computation
ARM Cortex-M4 multi-protocol radio SoC
Silicon Labs Wireless Gecko EFR32

Indoor Air Quality
Digital gas sensor
ams CCS811

Pressure
Absolute barometric pressure sensor
Bosch Sensortec BMP280

Motion Tracking
6-axis inertial measurement unit
TDK InvenSense ICM-20648

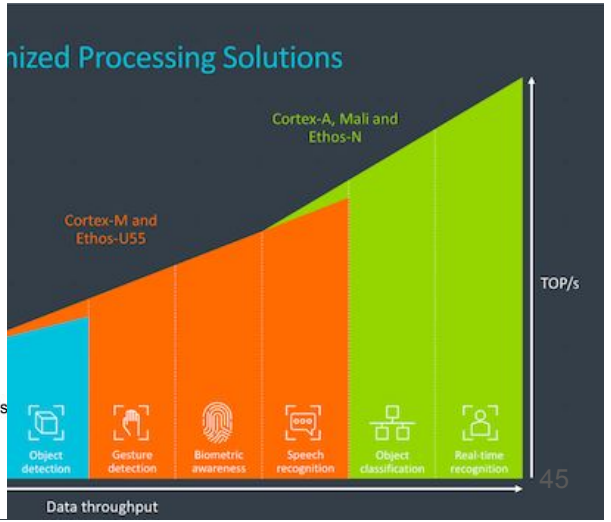


Audio Sensing
High quality MEMS microphone
TDK InvenSense ICS-43434

UV and Light
UV index and ambient light sensor
Silicon Labs Si1133

Temperature and Humidity
Relative humidity and temperature sensor
Silicon Labs Si7021

User Input and Feedback
2 push buttons and 4 high-power RGB LEDs



What's possible

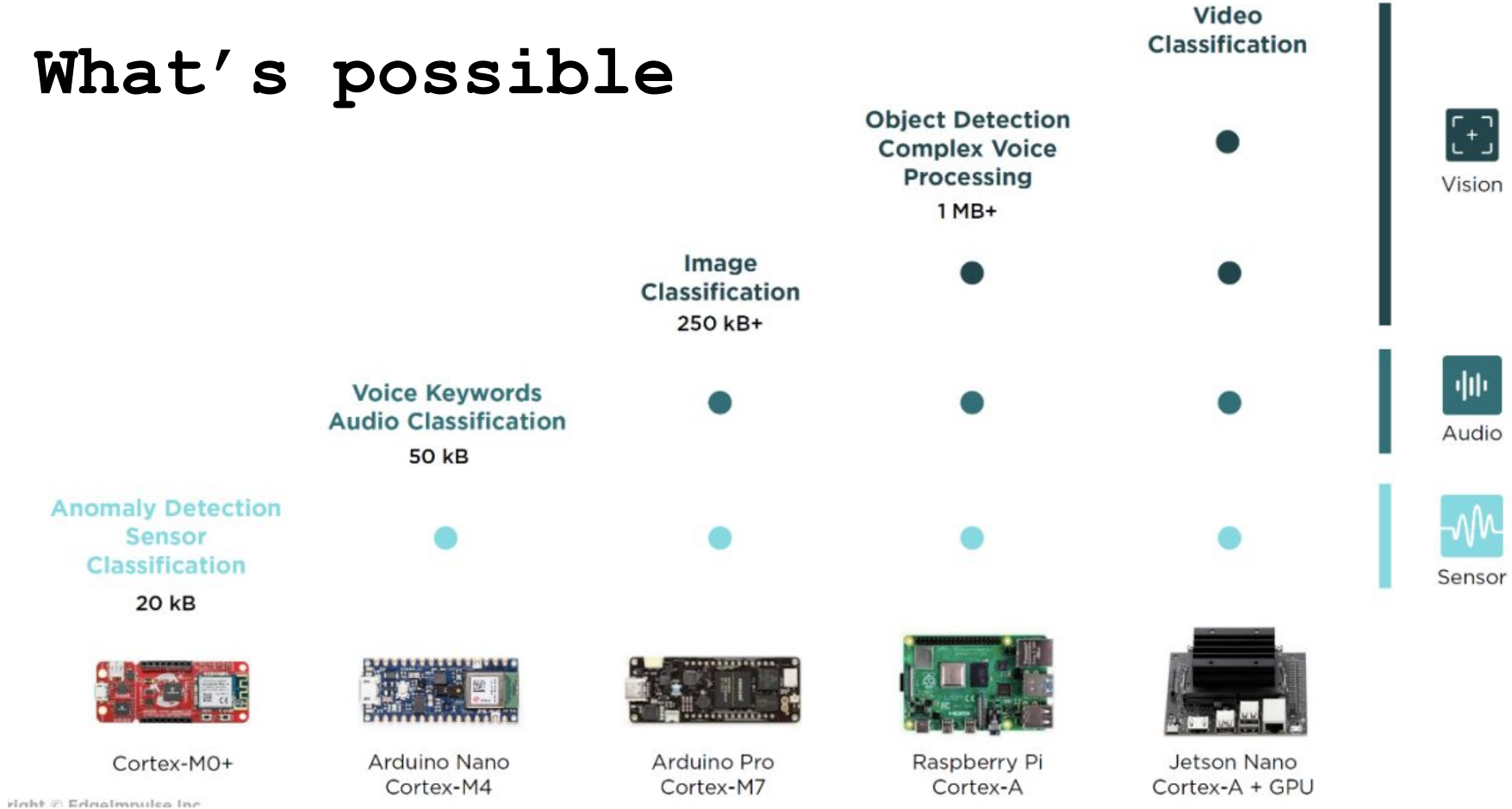


Figure 2. Typical development platforms and supported sensor types

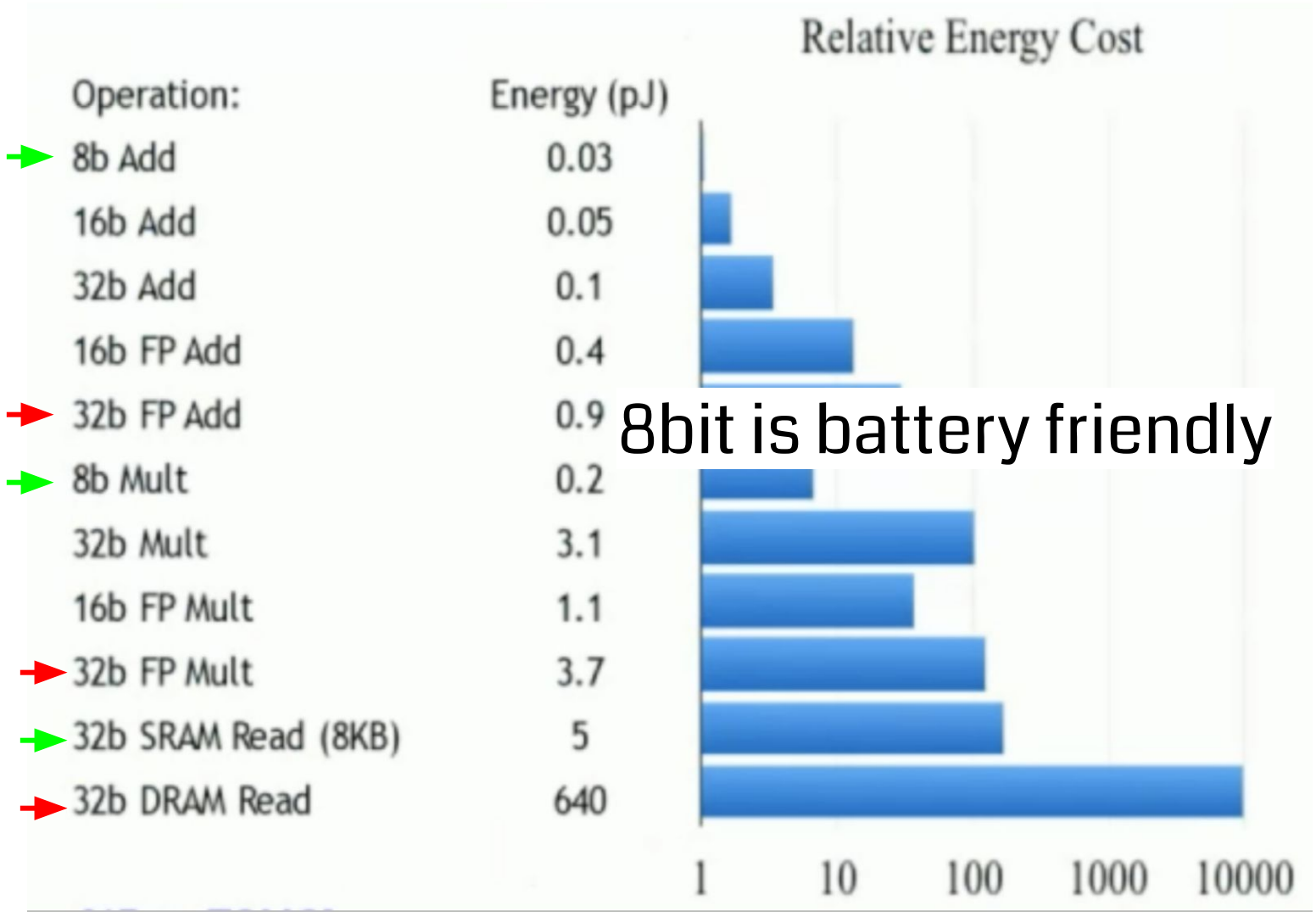
Source: Edge Impulse

Quiz

What is SIMD

Single Instruction Multiple Data
The Cortex-M4, Cortex-M7, Cortex-M33, Cortex-M35P, and Cortex-M55 processors provide SIMD instructions that operate on 8-or 16-bit integers. All registers are still 32-bits wide, but the SIMD instructions operate on 2 x 16-bit values or 4 x 8-bit values at the same time within a 32-bit register.

Relative energy cost of compute



8bit is battery friendly

An Example...

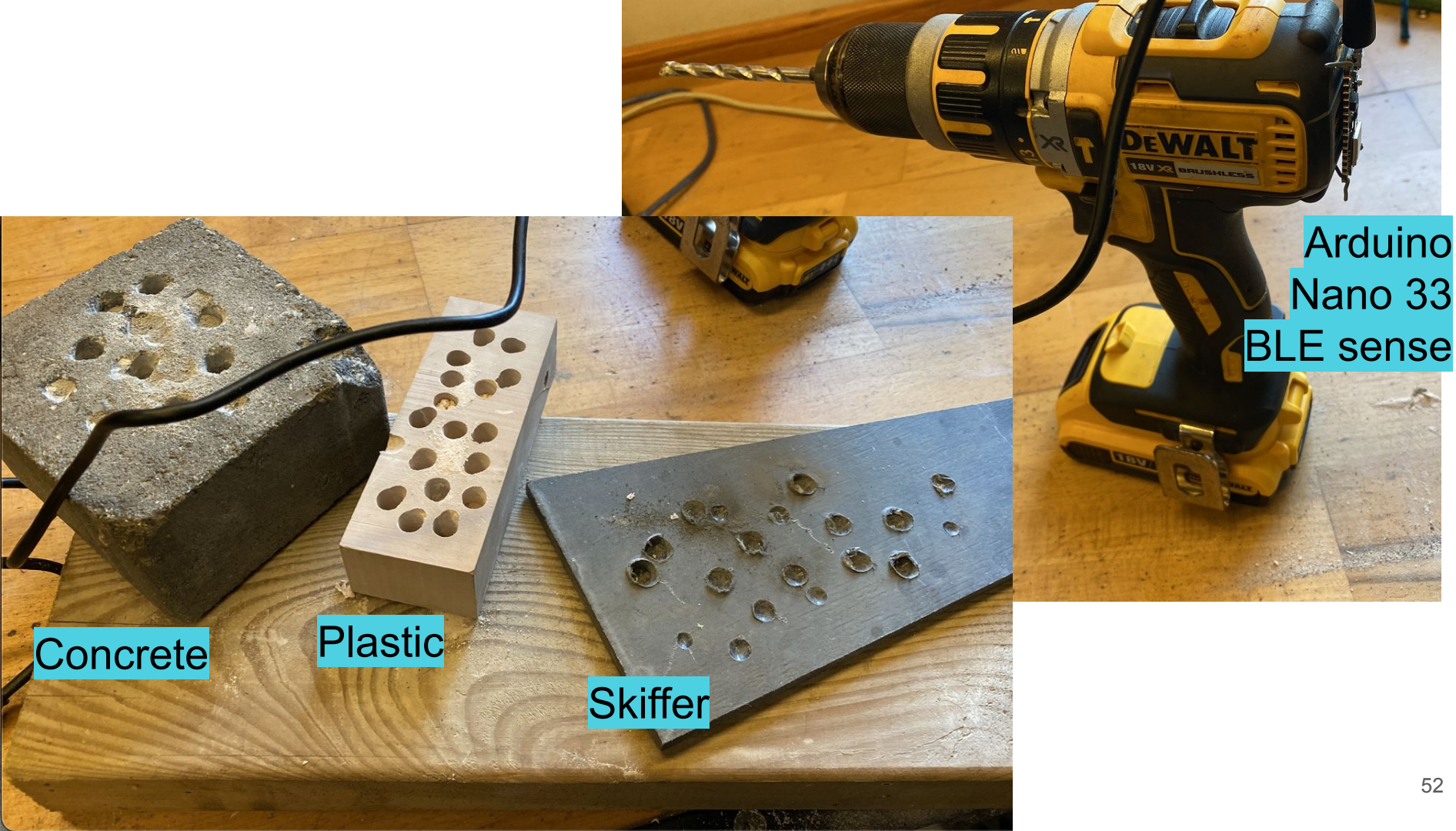
For example, the MobileNetV2 image classification network takes 22 million ops (each multiply-add is two ops) in its smallest configuration.

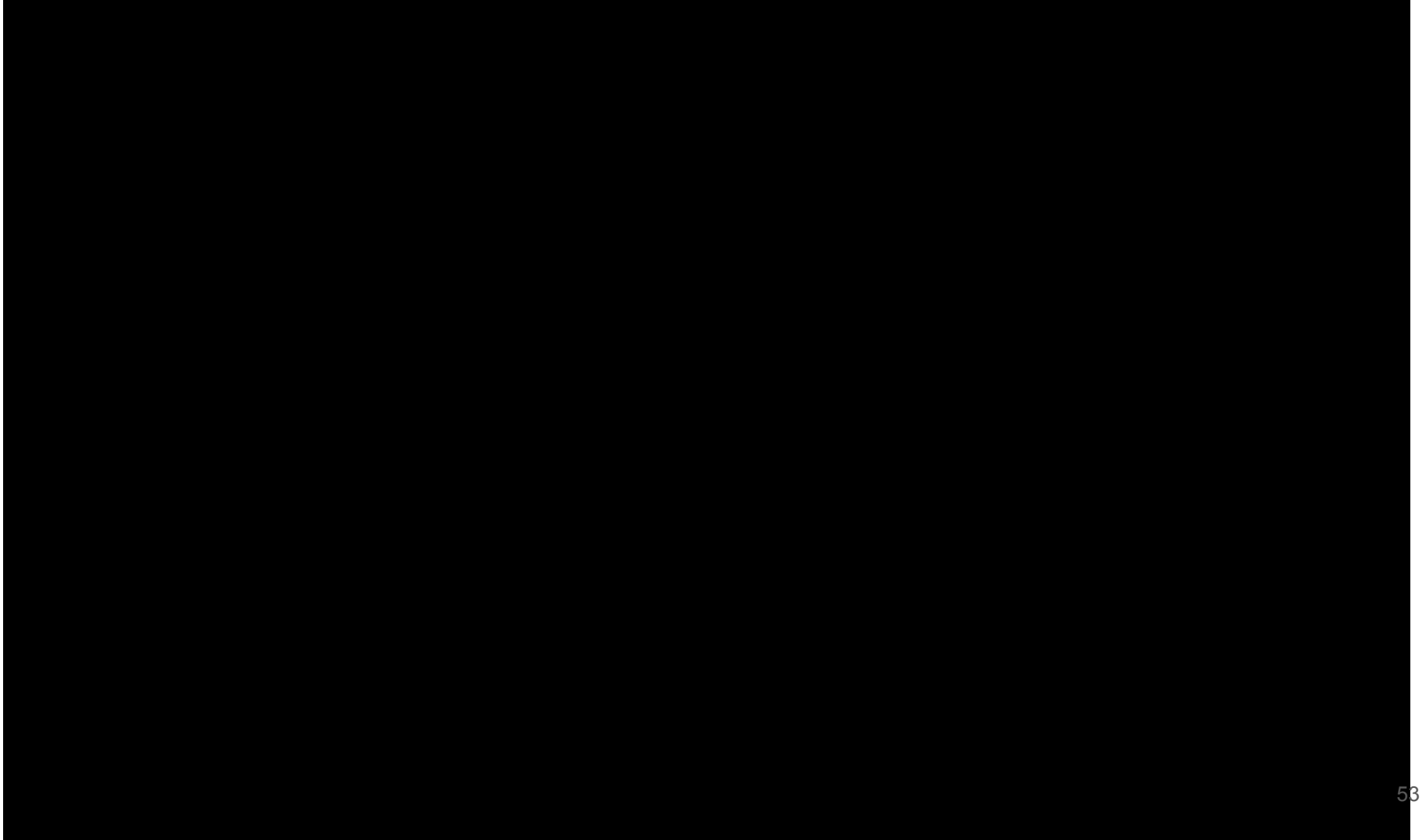
If I know that a particular system takes 5 picojoules to execute a single op, then it will take (5 picojoules * 22,000,000) = 110 microjoules of energy to execute. If we're analyzing one frame per second, then that's only 110 microwatts, which a coin battery could sustain continuously for nearly a year.

https://petewarden.com/2018/06/11/why-the-future-of-machine-learning-is-tiny/

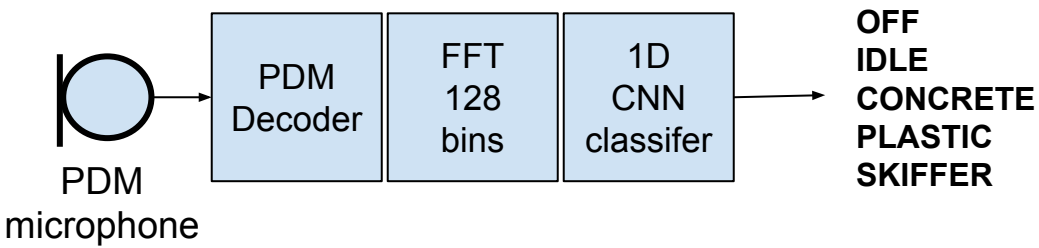
Another Example...

What are you drilling ?


Arduino
Nano 33
BLE sense
Concrete
Plastic
Skiffer

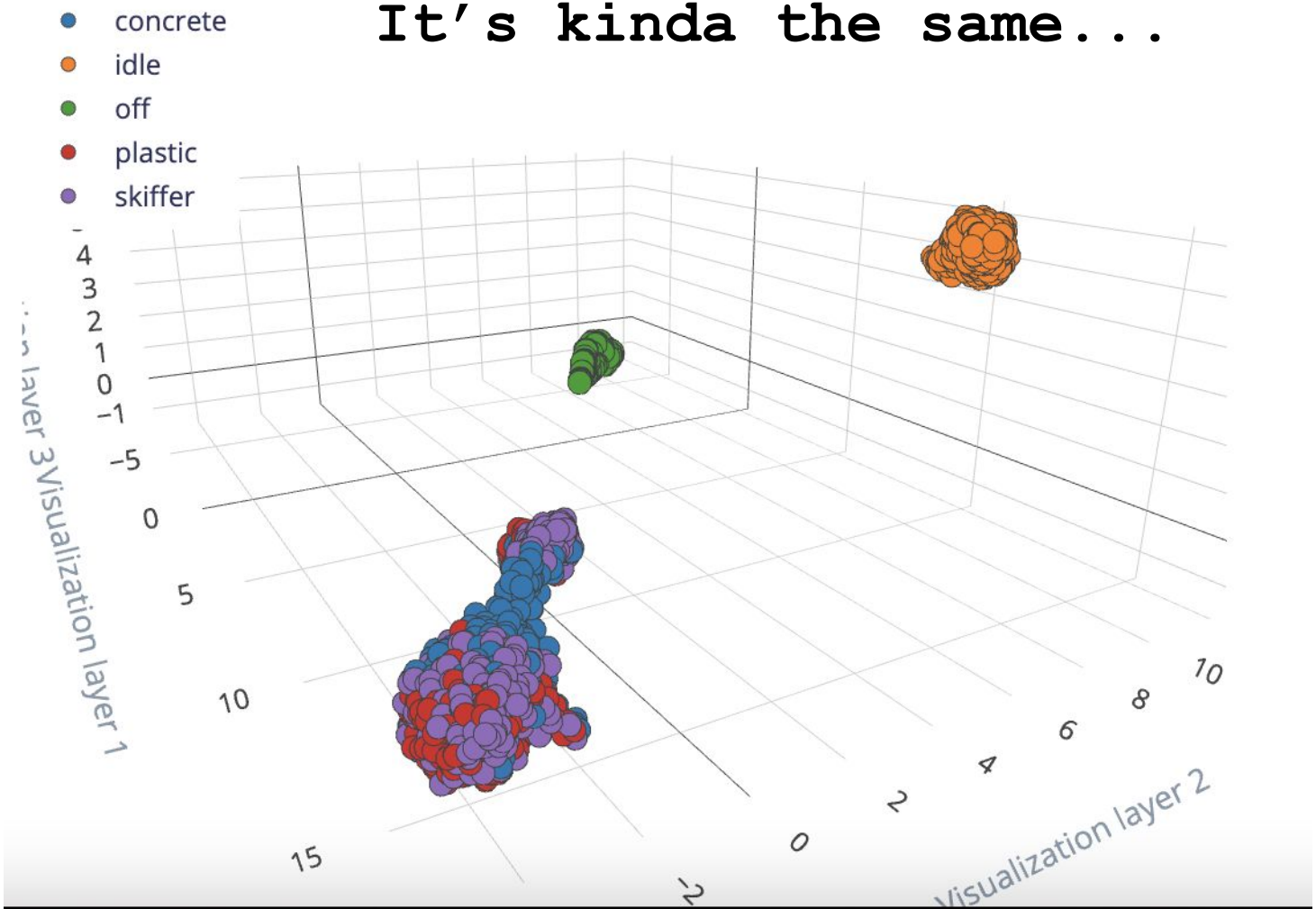


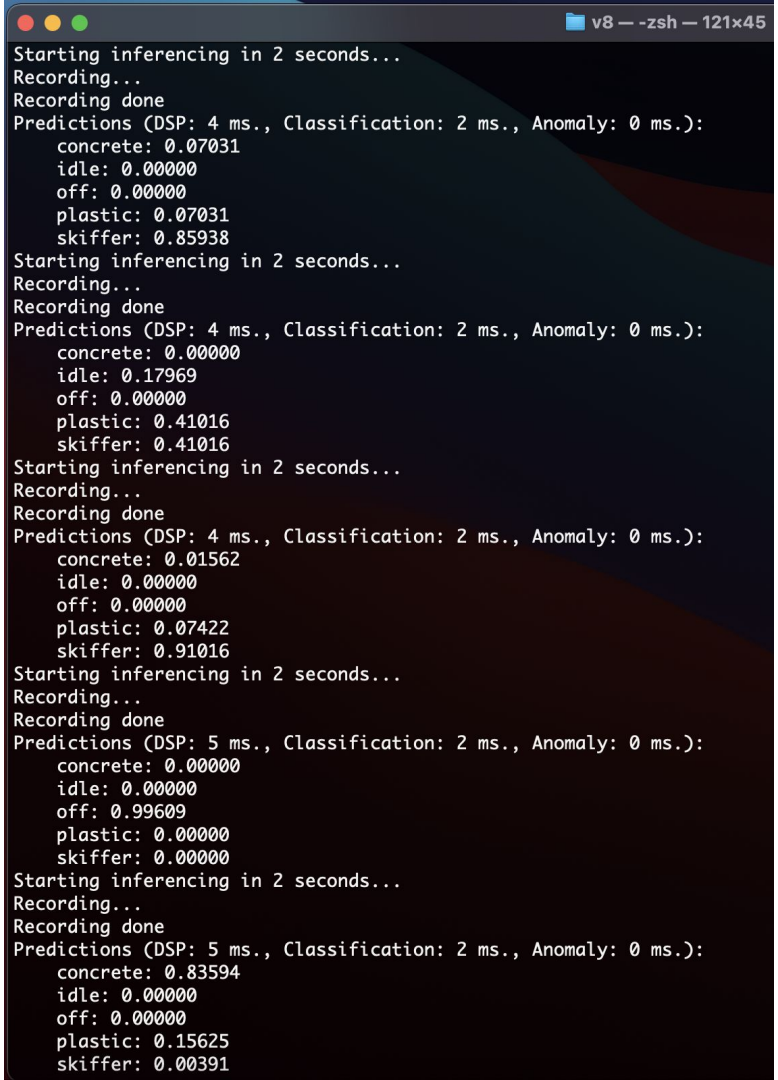
ML model



~6ms for signal processing + CNN

It's kinda the same...





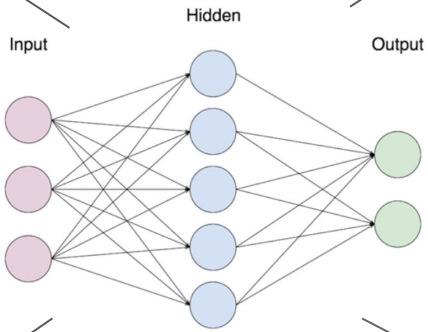
$$\sum_{n=0}^{N\_class} = 1.0$$

What is TinyML
- MachineLearning
-- ML vs AI
-- Learning vs. inference
-- ML vs Sequential programming
-- 5 min hands on
- TinyML
-- Motivation
-- Example use cases
-- Technological enablers
-- Hardware
-- Demo / example
-- Models - big to small
-- Software
-- Tools
-- 5 min hands on
- Resources
- Join
- Q&A

slides:https://github.com/opprud/tinyML/

Need for Size Reduction of Models

Power constraints:
Traditionally neural networks require massive amount of computational power and energy while running on CPU and GPUs

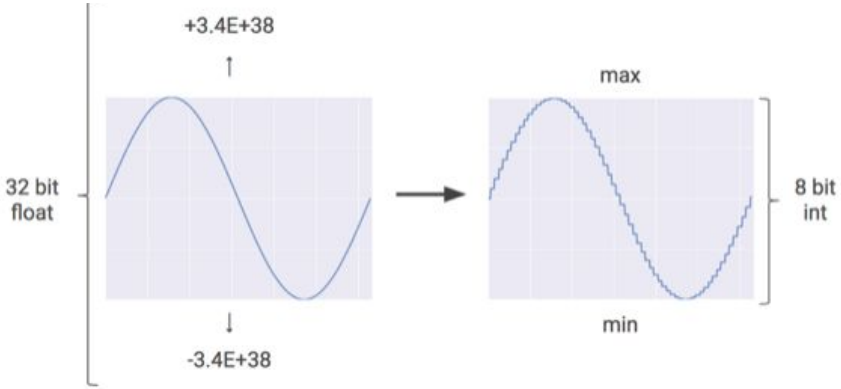


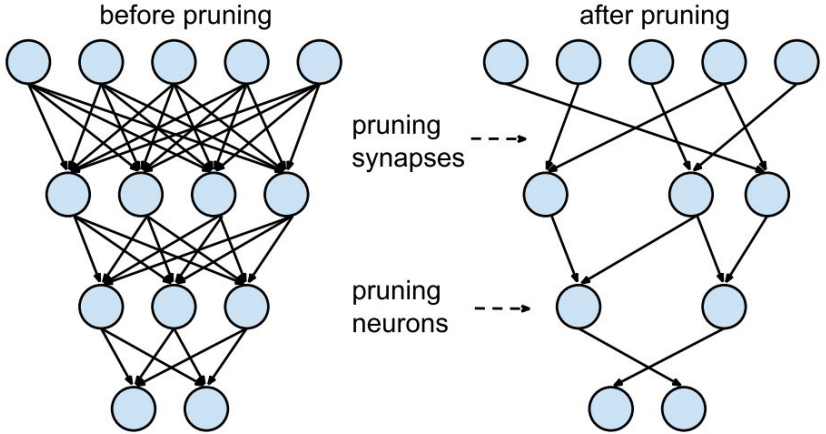
Composed of Floating Point Values: Neural Networks were traditionally trained to preserve accuracy and speed

Memory Constraints:
Laptops and PC come with at least 4GB of RAM whereas Microcontrollers has upto 256K of RAM / 1M Flash

Inference Efficiency:
We need model that takes less time for inference

Quantization and Pruning





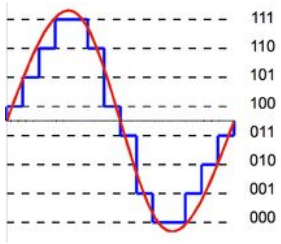
Quantization forces the network to simulate in lower bits

Pruning forces zero interconnection

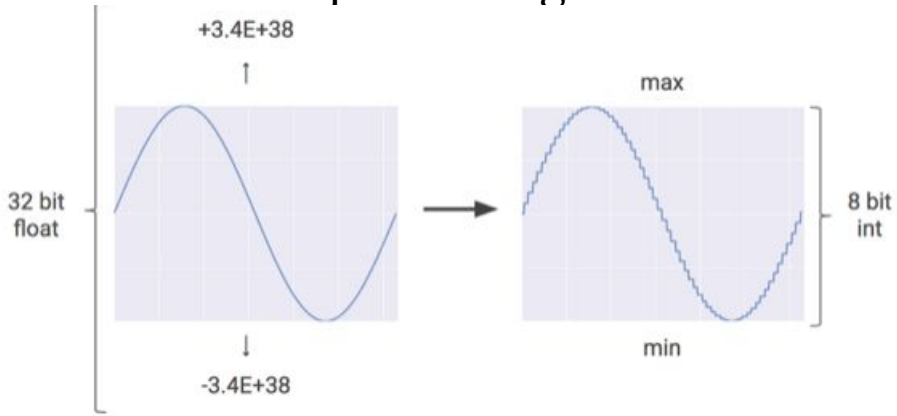
https://docs.google.com/presentation/d/1FdtR2o22HzH6iDJfNgTi11bhpMgZc5E-VpCleKt8vFE/

Definition of Quantization

Actual Definition: It is the process of mapping of input values to a large set of output values in a smaller set, with a finite number of elements; used in mathematics and signal processing.



Definition used for Edge Computing: It is the process of reducing the FP-32-bit values to lower bit precision values while preserving the information in the neural networks



How does INT8 Quantization work?

For eg, your neural networks has weight values from -10.00 to. 30.00

- Define Max and Min values
- While doing INT8 quantization, the least value = 0
- The maximum value = 255. ; since 2^8=256
- All the values in between are scaled to the range of 0 to 255

Quantized	Float
0	-10.0
255	30.0
128	10.0

What happens when we want to represent value = 10.00?

oldRange = oldMax - oldMin

newRange = newMax - newMin

newValue = ((oldValue - oldMin) * newRange / oldRange) + newMin

oldRange = 30-(-10) = 30+10 = 40

newRange = 255-0 = 255

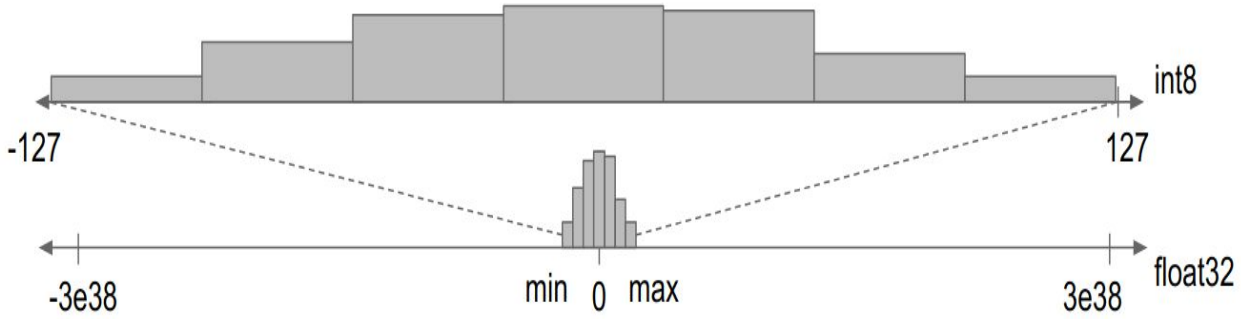
newValue = ((10-(-10))*255/40)+0

= 127.5~rounding off to 128

Tensorflow Optimization Toolkit

1. It contains tools for
    a. Quantization Aware Training
    b. Model Pruning
    c. Post Training Quantization
2. Choose layers you want to quantize in the model
3. Pair the reduced-precision operation definitions in the resulting model with kernel implementations that use a **mix of fixed- and floating-point math.**
4. The most sensitive ones with higher precision, thus typically resulting in little to no final accuracy losses for the task, yet a significant speed-up over pure floating-point execution

Types of Quantization



1. In float 32 you have values from -1 to +1, but in int 8 you are converting it to 256 numbers
2. Weight Quantization
3. Weight and Activation Quantization

Types of Quantization- Choices Available

Technique	Benefits	Hardware
Dynamic range quantization	4x smaller, 2x-3x speedup	CPU
Full integer quantization	4x smaller, 3x+ speedup	CPU, Edge TPU, Microcontrollers
Float16 quantization	2x smaller, GPU acceleration	CPU, GPU

Types of Quantization- Choices Available

Type of Quantization	Quantized parts of the neural networks
Dynamic Range Quantization:The simplest form of post-training quantization statically quantizes only the weights from floating point to integer, which has 8-bits of precision:	You want to **reduce latency,** so on training, you focus on quantizing the **weights** At inference, weights are converted from 8-bits of precision to floating point and computed using floating-point kernels. This conversion is done once and cached to reduce latency.
Full Integer Quantization: by making sure all model math is integer quantized. And you can run it on hardware accelerators	Reduction of latency, memory usage, Entire model math is quantized, weights and activations
Float16 Quantization: You can reduce the size of a floating point model by quantizing the weights to float16, the IEEE standard for 16-bit floating point numbers. To enable float16 quantization of weights	• It reduces model size by up to half (since all weights become half of their original size). • It causes minimal loss in accuracy. • It supports some delegates (e.g. the GPU delegate) which can operate directly on float16 data, resulting in faster execution than float32 computations.

Quantization Aware Training~Mimicking Real World Quantization at Inference

1. Enables you to train and deploy models with the performance and size benefits of quantization, while retaining close to their original accuracy
2. Problems with quantization- Recaps and losses:
    a. process of going from higher to lower precision is lossy in nature.
    b. You are rounding off all values: For example, all values in range [2.0, 2.3] may now be represented in one single bucket. This is similar to rounding errors when fractional values are represented as integers.
    c. Accumulation of loss: Neural Networks consist of MAC operations, several multiply-add computations, these losses accumulate
    d. More computational error

Post training optimized example, CNN

Available optimizations for Neural Network (Keras)

Quantized (int8)

⭐

Currently selected

This optimization is recommended for best performance.

RAM USAGE	LATENCY
4.5K	3 ms
FLASH USAG	**ACCURACY**
44.6K	56.08%

CONFUSION MATRIX

66.4	0	6.8	3.6	7.6	15.6
0	99.2	0	0	0.4	0.4
0	0	100	0	0	0
17.0	0.4	2.2	24.3	10.3	45.7
13.7	0.2	4.0	5.6	33.6	43.0

Unoptimized (float32)

Click to select

RAM USAGE	LATENCY
5.1K	12 ms
FLASH USAG	**ACCURACY**
74.9K	55.47%

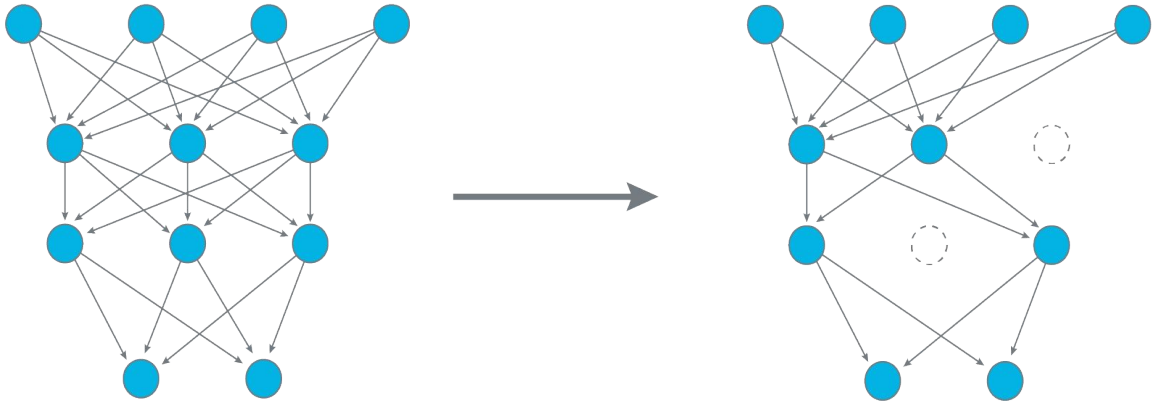
CONFUSION MATRIX

64.4	0	6.8	3.4	7.7	17.6
0	99.6	0	0	0.4	0
0	0	100	0	0	0
15.6	0.4	2.2	25.7	9.7	46.4
12.1	0.4	4.0	5.9	31.8	45.9

Try it out

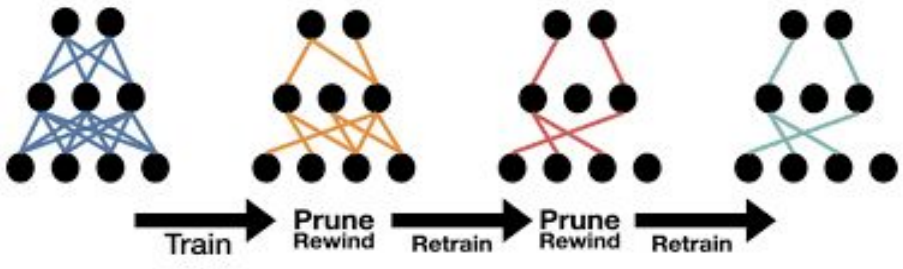
colab

Model Pruning



Model Pruning

- **Step 1:** Rank Weights
  - Layerwise or across the whole network
- **Step 2:** Remove Weights
  - Weights are removed by setting them to zero
- **Step 3:** Retrain your model
  - Fine tune your model to prevent drop in accuracy



Does Pruning actually help ?

- Saving a FP32 zero, also requires 32 bits
  - No memory savings
- Reading a FP32 zero requires same time/effort as a non-zero number
  - No memory read/load savings
- Multiplying with 0 may or may not be optimized
  - Some execution savings....maybe?

“Why is TinyML so important, you ask?

Simple – because it’s tiny!”

https://www.seeedstudio.com/blog/2021/06/14/everything-about-tinyml-basics-courses-projects-more/

80/20

Steve Lohr of The New York Times said: “Data scientists, according to interviews and expert estimates, spend **50 percent to 80 percent of their time** mired in the mundane labor of collecting and preparing unruly digital data, before it can be explored for useful nuggets.”

https://medium.com/@gauravkth/why-data-preparation-is-an-important-part-of-data-science-e40e054f18eb

What is TinyML
- MachineLearning
-- ML vs AI
-- Learning vs. inference
-- ML vs Sequential programming
-- 5 min hands on
- TinyML
-- Motivation
-- Example use cases
-- Technological enablers
-- Hardware
-- Demo / example
-- Models - big to small
-- Software
-- Tools
-- Quiz
- Resources
- Join
- Q&A

slides:**https://github.com/opprud/tinyML/**

READM

TensorFlow Install Learn API Resources Community Why TensorFlow Search

SensiM

SensiM

Build rapid pro

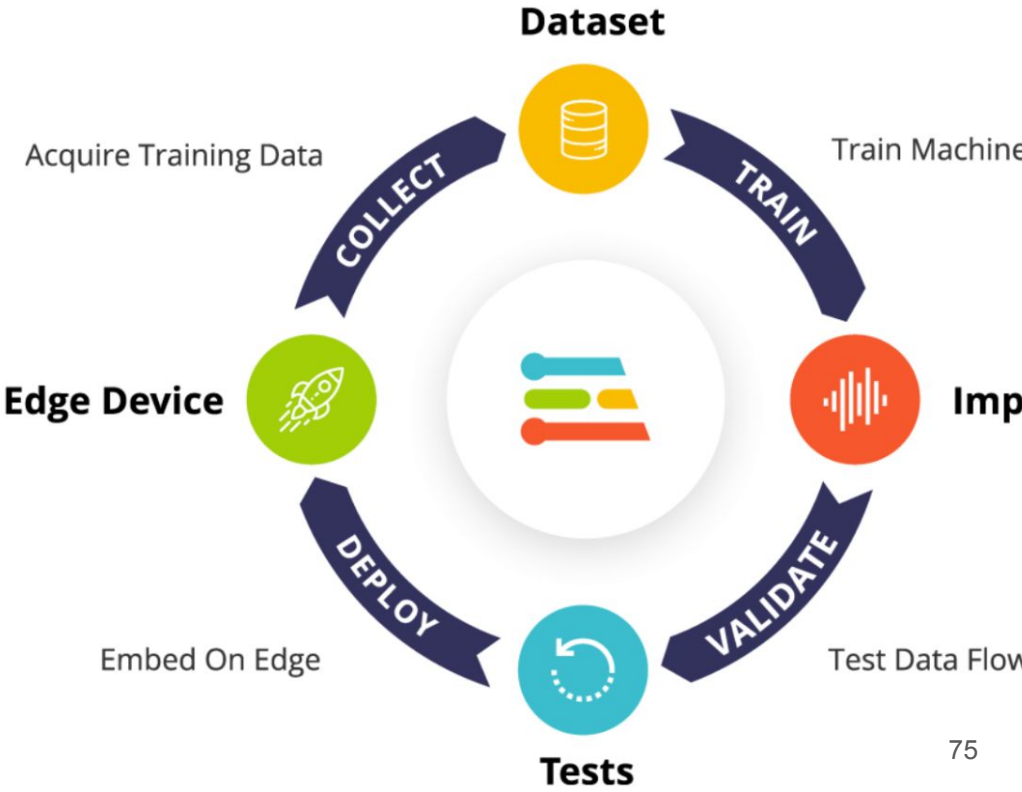
Data Capture

Why Choc

Raw Sign
Capture

Start using your device data

Enable valuable use of the 99% of sensor data discarded today due to cost, bandwidth or power. From getting started to MLOps in production, Edge Impulse provides maximum efficiency on a wide range of hardware from MCUs to CPUs thanks to Edge Optimized Neural (EON)™ technology.



github.com/emlearn/emlearn

github.com/eloquentarduino/micromlgen

README.md

Introducing MicroML

MicroML is an attempt to bring Machine Learning algorithms to microcontrollers. Please refer to this blog post to an introduction to the topic.

Install

```
pip install micromlgen
```

Supported classifiers

`micromlgen` can port to plain C many types of classifiers:

- DecisionTree
- RandomForest
- XGBoost
- GaussianNB
- Support Vector Machines (SVC and OneClassSVM)
- Relevant Vector Machines (from `skbayes.rvm_ard_models` package)
- SEFR
- PCA

```
from micromlgen import port
from sklearn.svm import SVC
from sklearn.datasets import load_iris
```

arm

Products

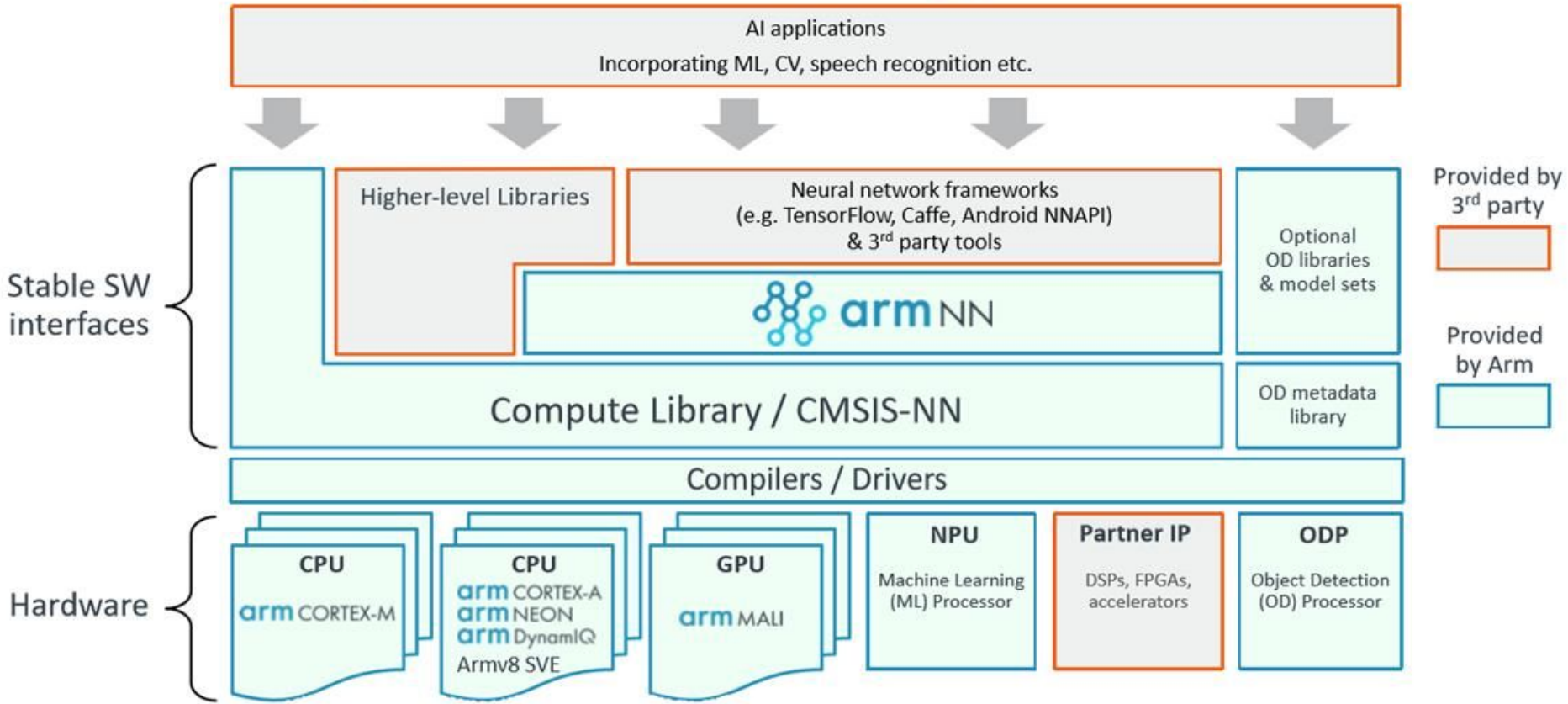
Solutions

Why Arm

Support & Training

Resources

Arm's heterogeneous Machine Learning platform



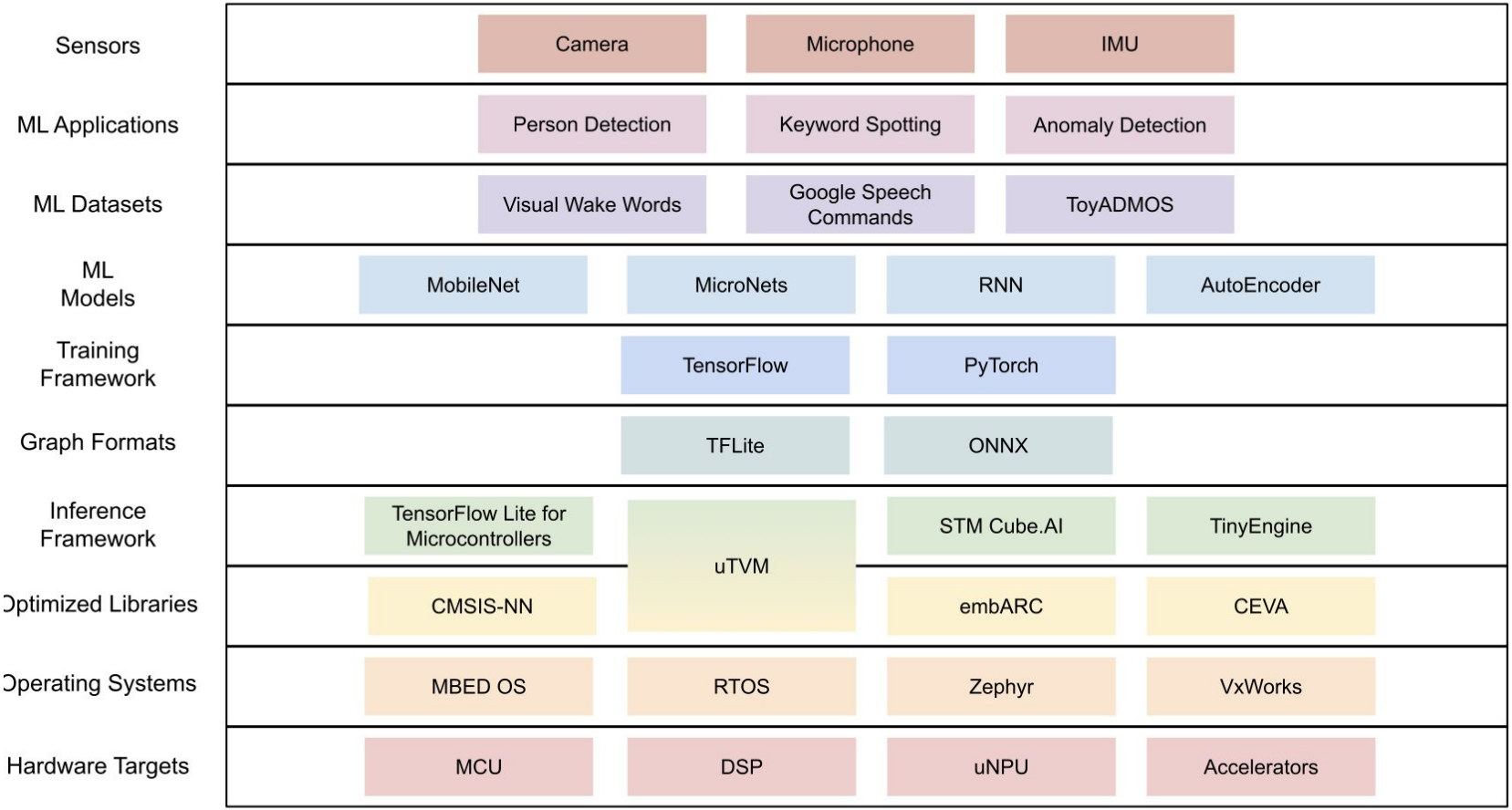


Figure 1: Summary of the Tiny Machine Learning Stack. There is diversity at every level which makes standardization for benchmarking challenging.

https://arxiv.org/abs/2106.07597

Resources

Benchmarking

- EEMBCs EnergyRunner
- MLPerf

Books and Articles

- TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers
- Model Quantization Using TensorFlow Lite
- What is TinyML?
- TinyML as-a-Service: What is it and what does it mean for the IoT Edge?
- TinyML is breathing life into billions of devices
- TinyML as a Service and the challenges of machine learning at the edge
- Predictions for Embedded Machine Learning for IoT in 2021
- Matthew Mattina: Life-Saving M
- How predictive maintenance is
- Tiny four-bit computers are nov
- Taking Back Control
- How AI is Taking on Sensors
- MLCommons™ Releases MLPer
- TapLock - A bike lock with mac
- Neural network architectures fo microcontrollers
- A natively flexible 32-bit Arm m
- Wearable Devices Can Reduce Collision Risk in Blind and Visually Impaired People
- TinyML in MicroCosmos

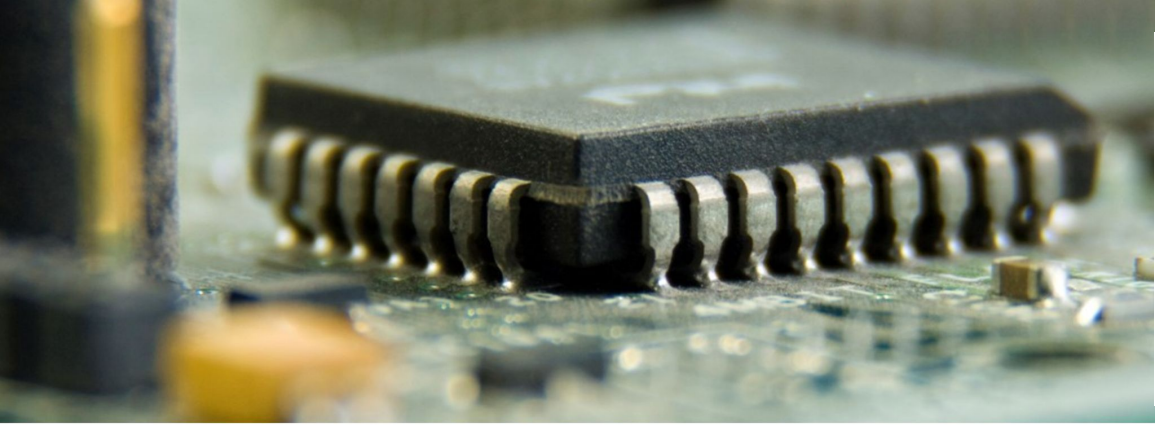
Courses

- CS249r: Tiny Machine Learning |Youtube
- Tiny Machine Learning (TinyML)| [code]
- Introduction to Embedded Machine Learning
- Embedded and Distributed AI course at Jonkoping University, Sweden
-

- TinyML: Analysis of Xtensa LX6 microprocessor for Neural Network Applications by ESP32 SoC | [pdf]
- LB-CNN: An Open Source Framework for Fast Training of Light Binary Convolutional Neural Networks using Chainer and Cupy | [pdf]
- Only Train Once: A One-Shot Neural Network Training And Pruning Framework | [pdf]

- TinyML using different frameworks applied to STM32F407 uC
- ESP32 Cam and Edge Impulse
- TinyML ESP32-CAM: Edge Image classification with Edge Impulse

Join



IDA Embedded

IDA Embedded er et socialt og fagligt netværk, der følger og aktivt deltager i udviklingen på det datatekniske fagområde.

Tilmeld

TINY ML

meetup COPENHAGEN

tinyML

4.87K subscribers

SUBSCRIBED

HOME VIDEOS PLAYLISTS COMMUNITY CHANNELS ABOUT

Uploads PLAY ALL

SORT BY

EMEA 2021 Success Stories - Spike-based neuromorphi…

344 views • 1 week ago

EMEA 2021 Success Stories - STMicroelectronics' AI…

162 views • 1 week ago

EMEA 2021 Success Stories - tinyML research under th…

109 views • 1 week ago

EMEA 2021 Partner Session Qualcomm

180 views • 2 weeks ago

EMEA 2021 Lightning Talks: A deeply embedded radar…

136 views • 2 weeks ago

EMEA 2021 Lightning Talks: A low power and High…

139 views • 2 weeks ago

EMEA 2021 tiny Talks: Building Heterogeneous…

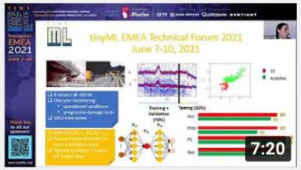
137 views • 2 weeks ago

tinyML Talks - Vikram Shrivastava : Dedicated…

280 views • 2 weeks ago

tinyML Talks India: DNN based AI application…

144 views • 2 weeks ago

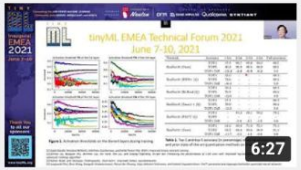


EMEA 2021 Student Forum: TinyML meets vibration-…

482 views • 3 weeks ago

EMEA 2021 Student Forum: Runtime DNN Performance…

71 views • 3 weeks ago



EMEA 2021 Student Forum: Squeeze-and-Threshold…

44 views • 3 weeks ago

Sponsors & Partners Make It Possible!

What is TinyML
- MachineLearning
-- ML vs AI
-- Learning vs. inference
-- ML vs Sequential programming
-- 5 min hands on
- TinyML
-- Motivation
-- Example use cases
-- Technological enablers
-- Hardware
-- Demo / example
-- Models - big to small
-- Software
-- Tools
-- 5 min hands on
- Resources
- Join
- **Q&A**

slides:https://github.com/opprud/tinyML/

Ekstra slides

What is “Artificial Intelligence?”

John McCarthy coined the term “artificial intelligence” in 1956

“[AI] is the science and engineering of making intelligent machines, especially intelligent computer programs...Intelligence is the computational part of the ability to achieve goals in the world.” --John McCarthy, 2007

What is “Machine Learning?”

Arthur Samuel coined the term “machine learning” in 1959

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." --Tom Mitchell, 1997

What is “Deep Learning?”

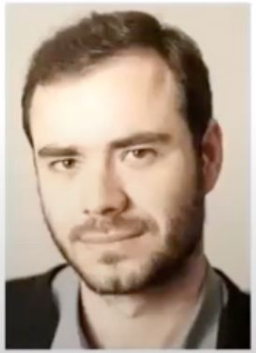
Rina Dechter coined the term “deep learning” in 1986

“Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher-level features from the raw input.” --Wikipedia

Differentiable programming / Software 2.0

*"Deep Learning est mort. Vive **Differentiable Programming**!*
*People are now building a new kind of software by assembling networks of **parameterized functional blocks** and by training them from examples using some form of **gradient-based optimization**.*
It's really very much like a regular program, except it's parameterized, automatically differentiated, and trainable/optimizable."
- Yann LeCun, Director of Facebook AI Research

*The "classical stack" of Software 1.0 is what we're all familiar with — it is written in languages such as Python, C++, etc. […] By writing each line of code, the programmer identifies a **specific point in program space with some desirable behavior**.*
*In contrast, Software 2.0 […] approach is to **specify some goal on the behavior of a desirable program** (e.g., "satisfy a dataset of input output pairs of examples", or "win a game of Go"), write a rough skeleton of the code (e.g. a neural net architecture) that identifies a **subset of program space to search**, and use the computational resources at our disposal to search this space for a program that works […] **the search process can be made efficient with backpropagation and stochastic gradient descent**.*
- Andrej Karpathy, Senior Director of AI at Tesla
  https://vimeo.com/274274744



Complex processing on small devices

About TinyML

TinyML is one of the fastest-growing areas of Deep Learning. In a nutshell, it's an emerging field of study that explores the types of models you can run on small, low-power devices like microcontrollers.

TinyML sits at the intersection of embedded-ML applications, algorithms, hardware and software. The goal is to enable low-latency inference at edge devices on devices that typically consume only a few **milliwatts** of battery power

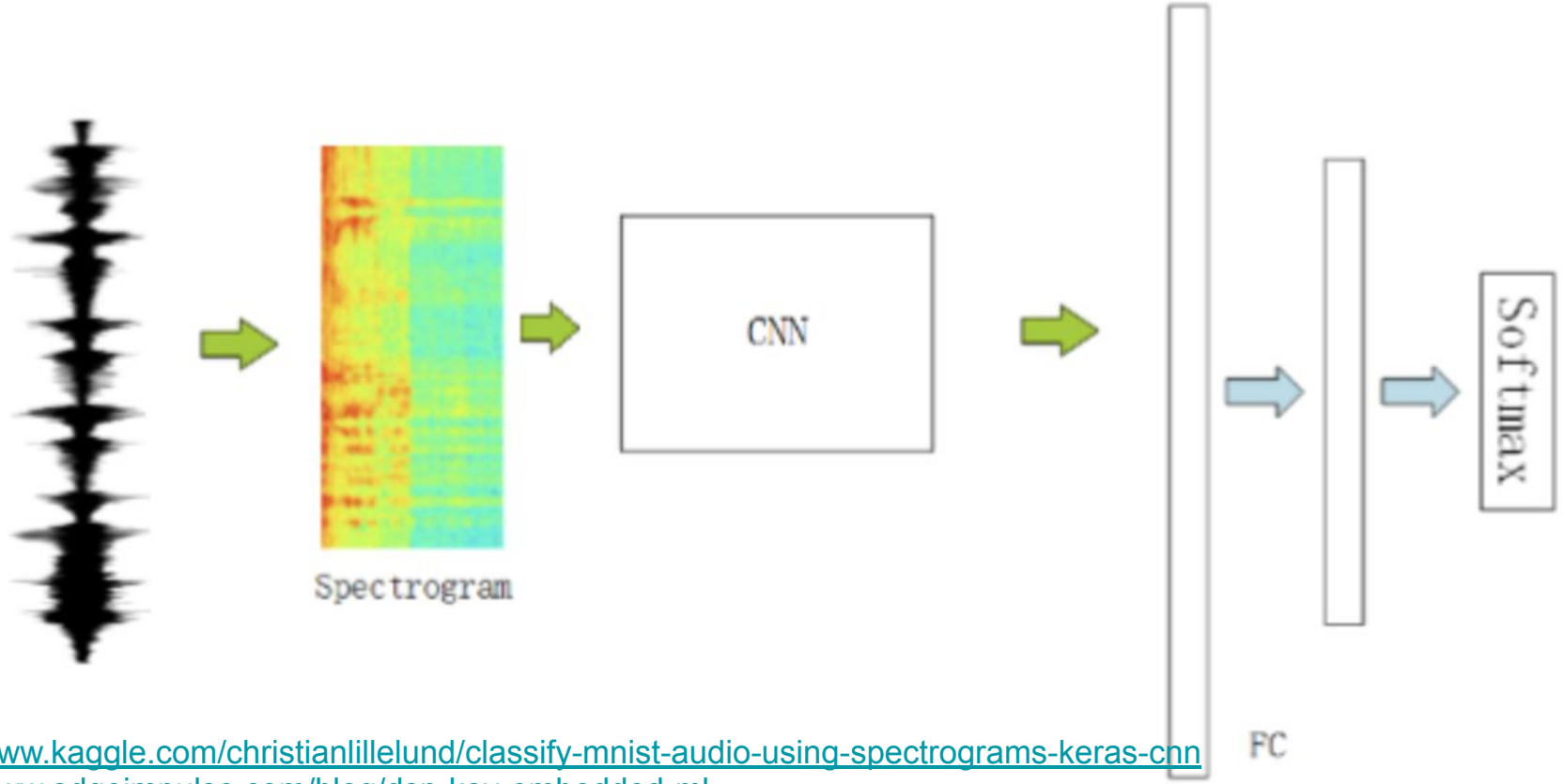
Complex processing on small devices

About TinyML

TinyML is one of the fastest-growing areas of Deep Learning. In a nutshell, it's an emerging field of study that explores the types of models you can run on small, low-power devices like microcontrollers.

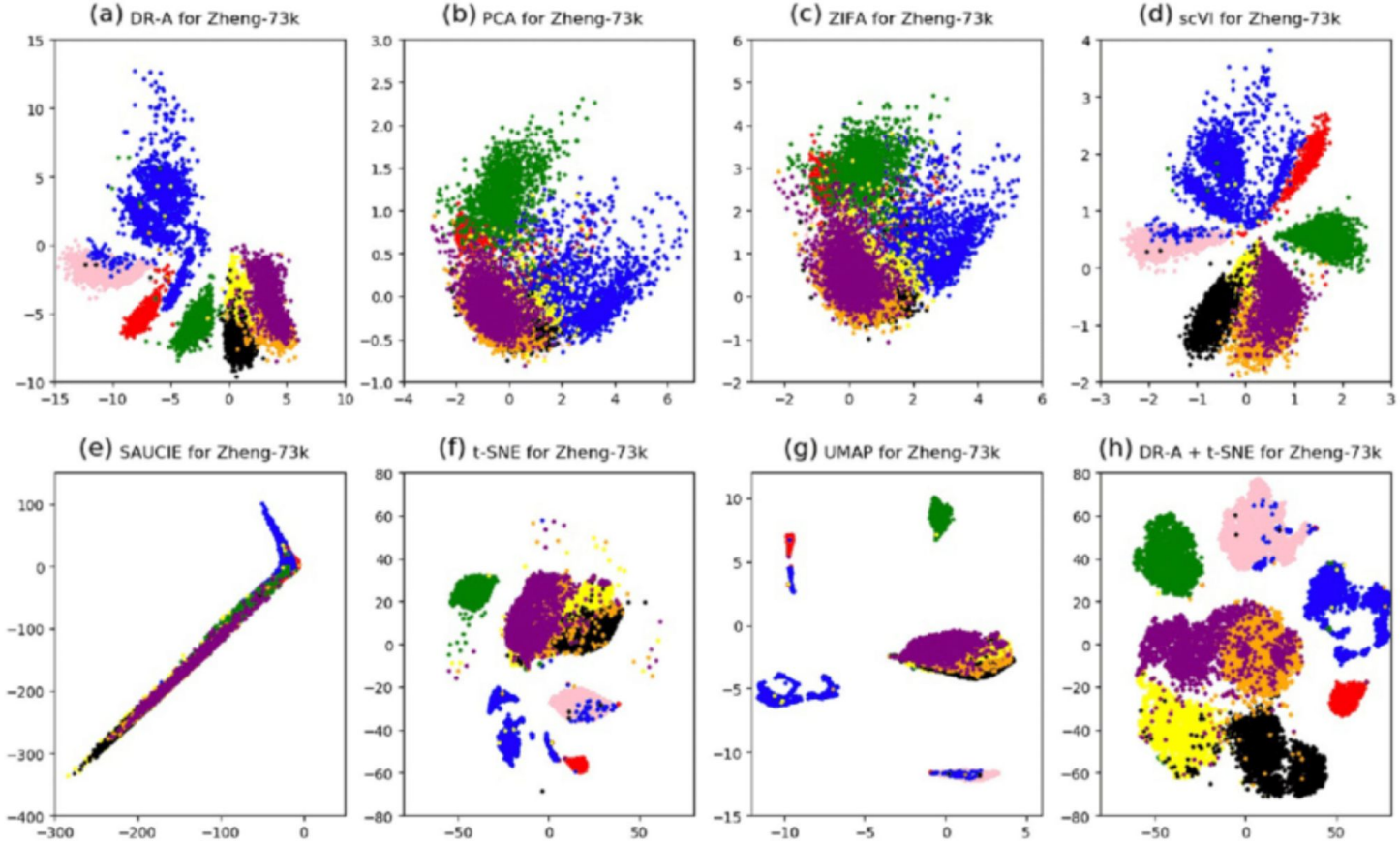
TinyML sits at the intersection of embedded-ML applications, algorithms, **hardware** and **software**. The goal is to enable low-latency inference at edge devices on devices that typically consume only a few **milliwatts** of battery power

ML ❤ Signal Processing



https://www.kaggle.com/christianlillelund/classify-mnist-audio-using-spectrograms-keras-cnn
https://www.edgeimpulse.com/blog/dsp-key-embedded-ml

Dimensionality reduction



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

Outcome, HW requirements (from SW)

- Flash for storing firmware and coef. 512Kb or more
- RAM for calculations, 256Kb or more
- Low power CPU, ARM preferred Cortex M4 or better w FPU/DSP instructions
- On chip RAM & Flash

CPU - candidate

Why the Ambiq Micro Apollo 3 Blue?

- Ambiq continues to bring new ultra low power microcontrollers (MCUs) to market
- From the Apollo 2 to the Apollo 3, Power efficiency has improved
- Apollo 3 is fabricated on TSMC's 40 nm eFlash LP process, but Ambiq's devices perform better than is typical for this process
- The addition of Bluetooth 5 (increased range and bandwidth) make this device even more compelling
- Very small die size @ 3.3 mm x 3.2 mm
- Design changes in transistor structure and layout as well as system architecture and power management demonstrate innovation that allows Ambiq to deliver high performance devices
  - Innovative designs observed in level shifter, flips flops and retention flip flops

Apollo2	Apollo2 Blue	Apollo3 Blue
48 MHz	48MHz	48MHz TurboSPOT™ 96MHz
32-bit ARM Cortex-M4F	32-bit ARM Cortex-M4F	32-bit ARM Cortex-M4F DMA
10uA/MHz	**10uA/MHz**	**6 uA/MHz**
1MB/256KB	1MB/256KB	1MB/384KB
1.8 - 3.6V	1.95 - 3.6V	1.8 – 3.6V
14 bit, 15-channel, up to 1.2 MSps	14 bit, 15-channel, up to 1.2 MSps	14 bit, 15-channel, up to 1.2 MSps
I²C /SPI master (6x) I²C /SPI slave UARTS (2)	I²C /SPI masters (4x) I²C /SPI slave UARTS (2)	I²C /SPI master (6x) I²C /SPI slave UARTS (2)
I2S slave for PDM Audio Pass-through	I2S slave for PDM Audio Pass-through	I2S slave for PDM Audio Pass-through
Dual Interface for Mono and Stereo Audio Microphones	Dual Interface for Mono and Stereo Audio Microphones	Dual Interface for Mono and Stereo Audio Microphones
-	BLE 5 w/ Dedicated Processor	BLE 5 w/ Dedicated Processor
-	-95dBm	-95dBm
-	+5dBm	+4dBm
-	3.5mA	3 mA
-	5.05mA @ 0dBm 8mA @ +5dBm	3 mA @ 0dBm
- 2.5 x 2.5 mm 49-pin CSP w/ 34 GPIO - 4.5 x 4.5 mm 64-pin BGA w/ 50 GPIO	4 x 4 x 0.9 mm 64-pin LGA with up to 31 GPIO	3.3 x 3.2 mm 65-pin CSP w/ 37 GPIO 5 x 5 mm 81-pin BGA w/ 50 GPIO