

# TinyML: Machine Learning meets the Internet of Things

Pietro Manzoni

- Universitat Politècnica de València (UPV)
- Valencia - SPAIN
- [pmanzoni@disca.upv.es](mailto:pmanzoni@disca.upv.es)

The GRC logo consists of a blue circle with the letters 'GRC' inside. To its right, the text 'GRUPO DE REDES DE COMPUTADORES' is written in a smaller font. Below the logo is a horizontal navigation bar with the following links: Home, Members, Papers, Projects, Teaching, Software, and Internal.

<http://grc.webs.upv.es/>



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA

The Networking Research Group (GRC - Grupo de Redes de Computadores) of the Universitat Politècnica de València (UPV) was founded in 2000 and it is mainly composed of researchers of the Computer Engineering Department (DISCA). It keeps strong bonds and collaborations with other researchers in the same area in Spain and in the rest of the world.

The group research efforts are focused on offering Data Communication Solutions for Mobile Systems. The main areas of application are:

- AIoT infrastructures for environmental sustainability
- Drone-based networks
- Efficient IoT infrastructures development
- Intelligent Transport Systems
- LPWAN-based networks
- Mobile edge computing
- Pub/Sub systems
- Social sensing



## Infos and News:

- [Overview of GRC research \[Sept. 2021\]](#)
- [GRC YouTube channel](#)
- [COVIDsensing: a tool to analize COVID spreading using AI](#)

## Events and CfPs:

### Conferences:

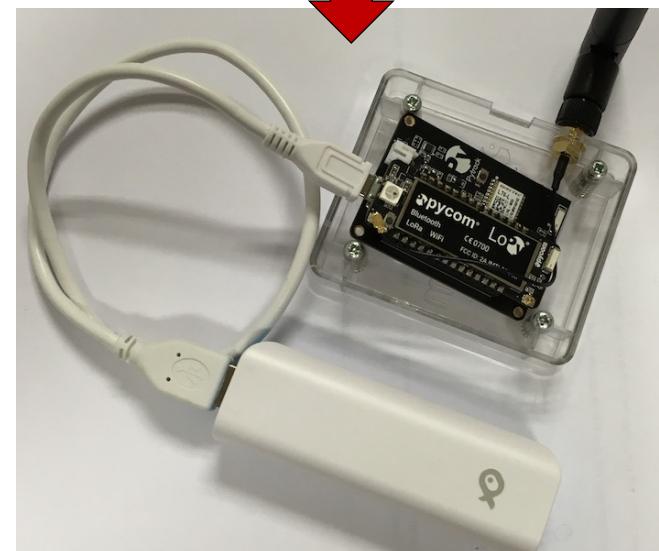
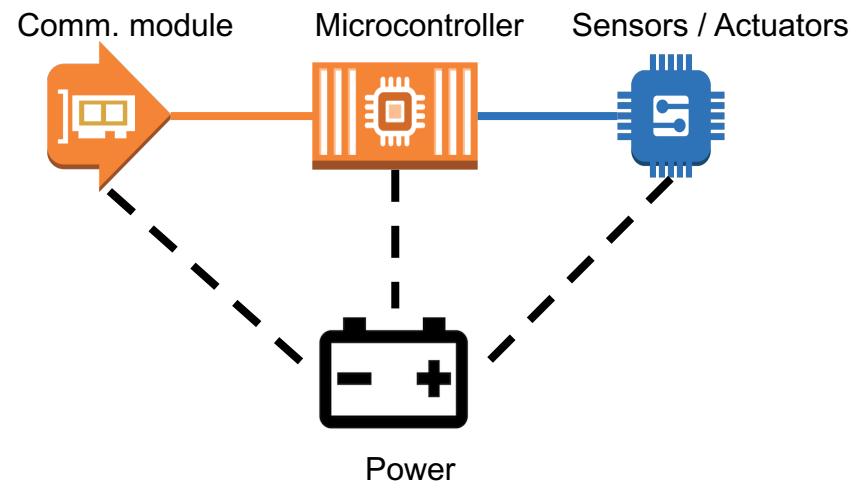
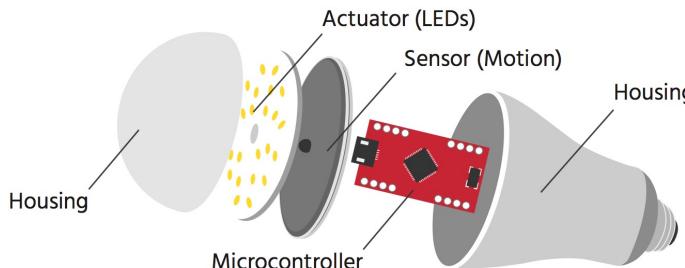
- [GoodIT](#), International Conference on Information Technology for Social Good, September 7-9, 2022, Limassol (CYPRUS).
- [NET4us](#), Workshop on Networked sensing systems for a sustainable society, during SIGCOMM 2022, August 22-26, 2022, Amsterdam (Netherlands).

### Journals Special Issues:

- Computer Networks, Elsevier, Special Issue on "[Pub/sub solutions for interoperable and dynamic IoT systems](#)". Submission deadline: to be opened soon.
- Frontiers, Loop, Research Topic on "[SDN migration challenges and practices in ISP/Telcos' Networks](#)". Submission deadline: open.
- ITU Journal, ITU, Special issue on "[Network virtualization, slicing, orchestration, fog and edge](#)

- The **Internet of Things** (IoT) refers to a system of interrelated, Internet-connected objects ("things"), that can collect and transfer data.
- **Machine Learning** is typically associated with computationally heavy cloud-based solutions with relatively high latencies, high power consumption, and the need for high bandwidths links.
- **With TinyML, IoT and ML come together by shrinking deep learning networks to fit on tiny hardware, thus requiring low energy, low bandwidth links, and reduced latencies.**

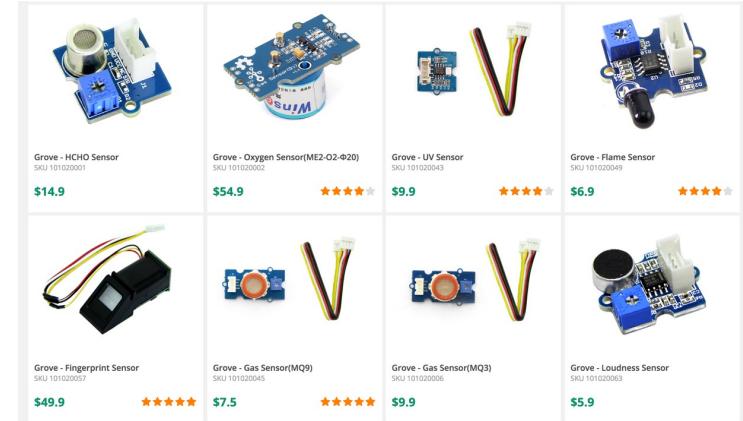
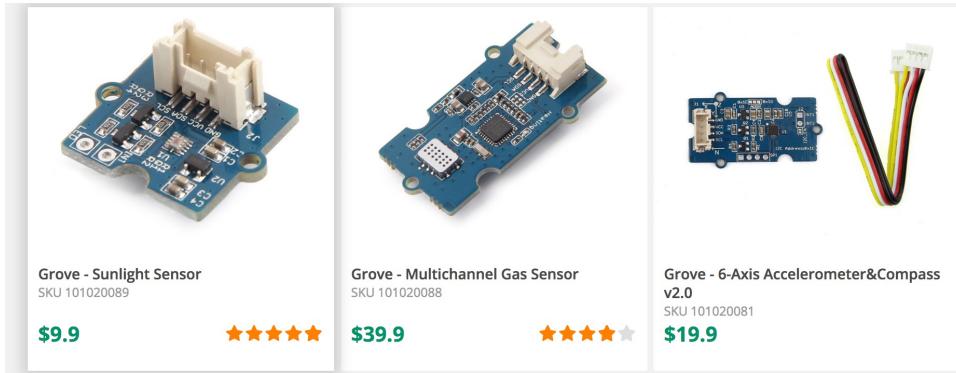
- A “thing” generally consists of **four main parts**:
  - Sensors & actuators
  - Microcontroller
  - Communication unit
  - Power supply
- A “thing” has the **following properties**:
  - It’s usually powered by battery. This implies limited source of energy.
  - It’s generally small in size and low in cost. This limits their computing capability.



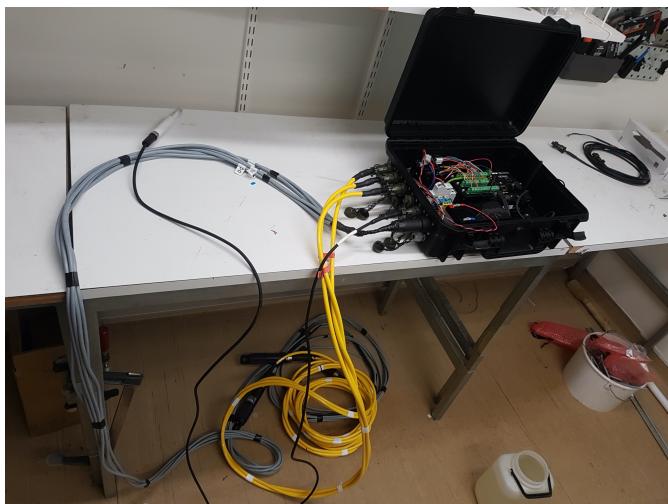
A Reference Guide to the Internet of Things Copyright © 2017 Bridgera LLC, RIoT

# Classic Sensors

cheap...



expensive...



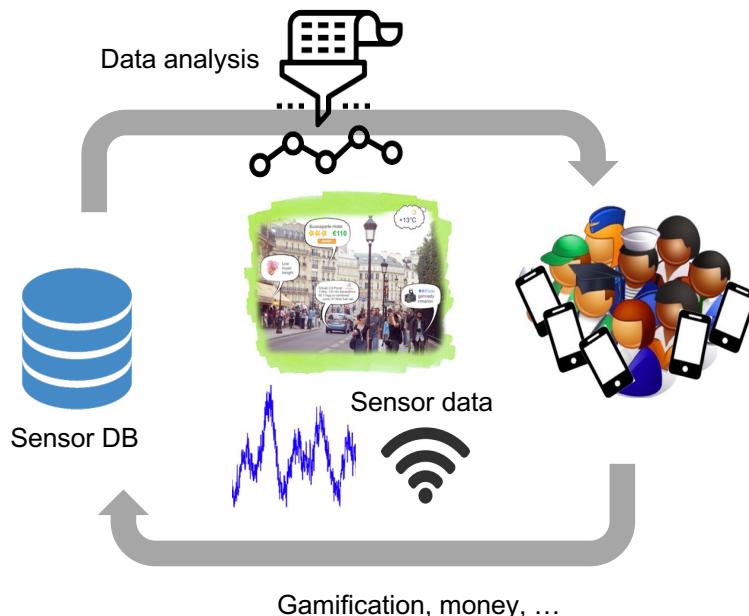
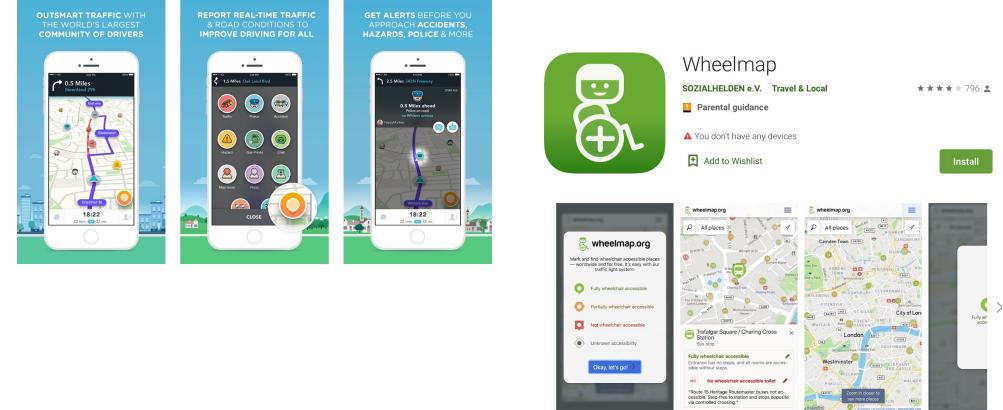
1.2 Oxygen Optode 4531 dimensions



Parameter	Output	Default range <sup>a)</sup>	Calibrated range	Accuracy	Resolution
Oxygen Concentration	0 - 5V	0 to 800µM	0 to 500µM	<8µM or 5% whichever is greater	<1µM
	4 - 20mA	0 to 800µM	0 to 500µM	<9µM or 5.2% whichever is greater	<1µM
Oxygen Saturation	0 - 5V	0 – 200%	0 - 120%	<5 %	<0.4%
	4 - 20mA	0 – 200%	0 - 120%	<5.2 %	<0.4%
Temperature	0 - 5V	-5 to + 35°C	0 - 36°C	±0.1°C	±0.01°C
	4 - 20mA	-5 to + 35°C	0 - 36°C	±0.15°C	±0.02°C

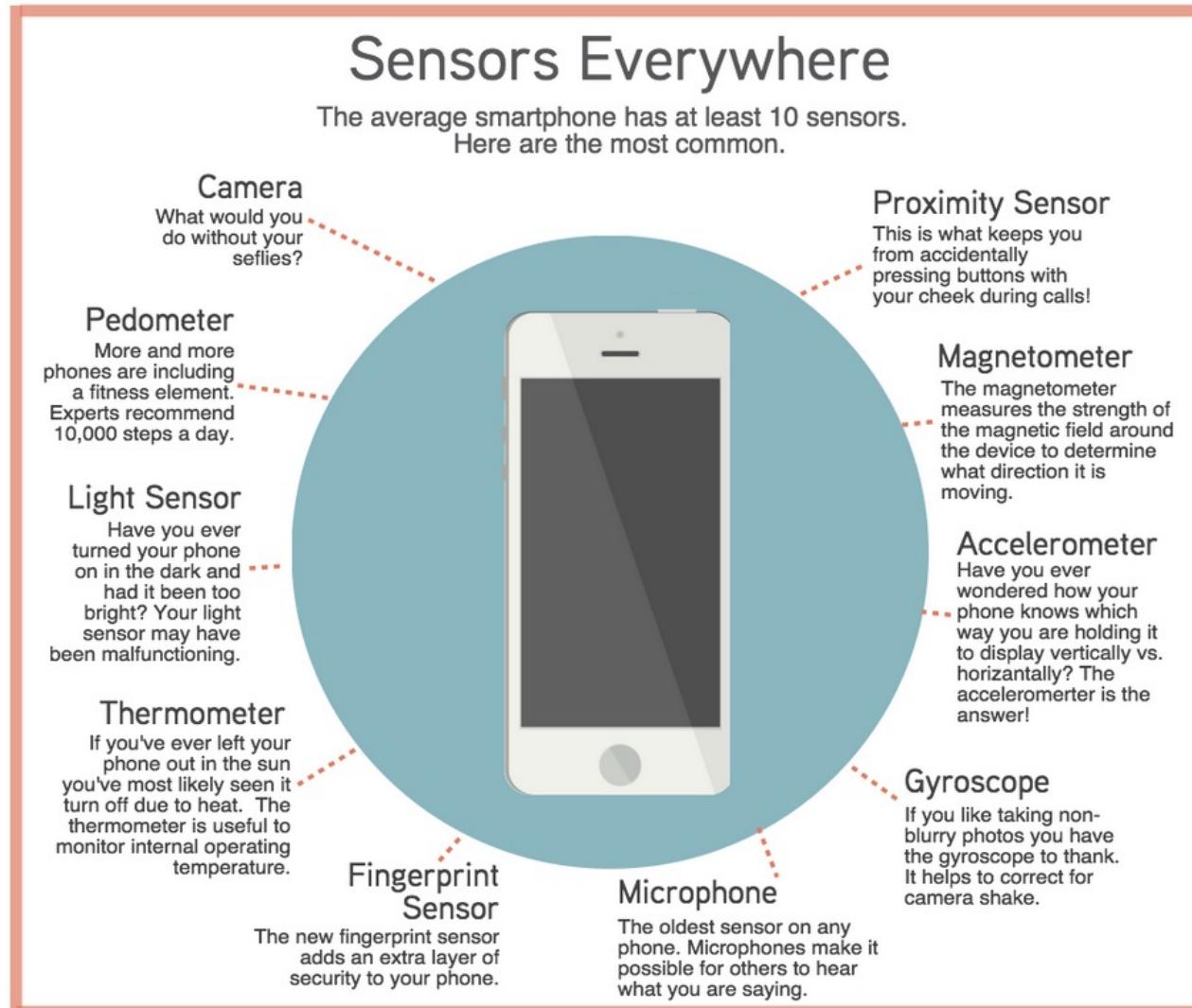
# Beyond conventional sensors: humans as sensors

- Social sensing: E.g., tweeting real-world data and/or events



**Crowdsensing, crowdsourcing**  
**Participatory crowdsensing**, where the users voluntarily participate in contributing information.  
**Opportunistic crowdsensing**, where the data is sensed, collected, and shared automatically without user intervention and in some cases, even without the user's explicit knowledge.

# Endpoints have sensors... tons of sensors



<https://github-wiki-see.page/m/Apt3kStudio/Phone/wiki/Sensors-on-Smartphones>

- Today, the **most common power source is a battery**, but there are several other possibilities for power, such as **solar cells, piezoelectricity, radio-transmitted energy, and other forms of power scavenging**.
- Rechargeable batteries are not particularly well-suited to smart objects.
  - Instead of using rechargeable batteries, battery-equipped smart objects are typically designed so a single battery should last the entire lifetime of the smart object.



LiPo



LiPo



Li-ion

Lithium Ion (Li-ion)  
Lithium-Ion Polymer (Li-Po)

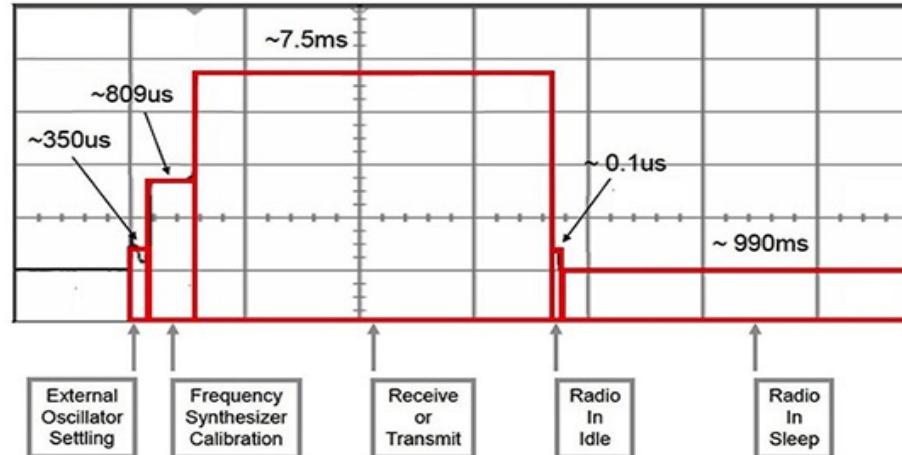
# How much power can you get from an AA?

- There are around 10,000 Joules in an AA battery.
- There are 31,536,000 seconds in a year.
- So, if you rely on one AA battery, in theory you've got an average 0.000317 watts of continuous power, or 317  $\mu$ W
- Actually in practice it's a lot more complex with battery characteristics, but a rule of thumb is that three AAs might give you about 1 mW of power for a year, if you're lucky.



# Communication Device

- Of the hardware components of a “thing”, the radio is usually the most power-consuming component.
  - Compared to the power consumption of the microcontroller or the sensors, the radio transceiver often uses ten times as much power.
- For low-power radios, only a small portion of the power consumption is used to send the radio signal into the air.
  - → listening is as power consuming as sending.



# Microcontroller: ESP32

- The ESP32 series employs a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations. ESP32 is created and developed by Espressif System
- CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS
- Memory: 520 KiB SRAM
- Wireless connectivity:
  - Wi-Fi: 802.11 b/g/n
  - Bluetooth: v4.2 BR/EDR and BLE
- Peripheral interfaces:
  - 12-bit SAR ADC up to 18 channels
  - 2 × 8-bit DACs
  - 10 × touch sensors (capacitive sensing GPIOs)
  - ...

<https://www.espressif.com/en/products/socs/esp32>

<http://esp32.net>



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

Power mode	Description			Power consumption
Modem-sleep	The CPU is powered on.	240 MHz*	Dual-core chip(s)	30 mA ~ 68 mA
		Single-core chip(s)	N/A	
	160 MHz*	Dual-core chip(s)	27 mA ~ 44 mA	
		Single-core chip(s)	27 mA ~ 34 mA	
	Normal speed: 80 MHz	Dual-core chip(s)	20 mA ~ 31 mA	
		Single-core chip(s)	20 mA ~ 25 mA	
Light-sleep	-			0.8 mA
Deep-sleep	The ULP coprocessor is powered on.			150 µA
	ULP sensor-monitored pattern			100 µA @1% duty
	RTC timer + RTC memory			10 µA
Hibernation	RTC timer only			5 µA
Power off	CHIP_PU is set to low level, the chip is powered off.			1 µA

# What does that all mean?



# So... Edge Computing

- A possible definition of **edge computing** is: “a part of a distributed computing topology in which information processing is located close to the edge — where things and people produce or consume that information.”

## Edge-to-cloud architecture layers

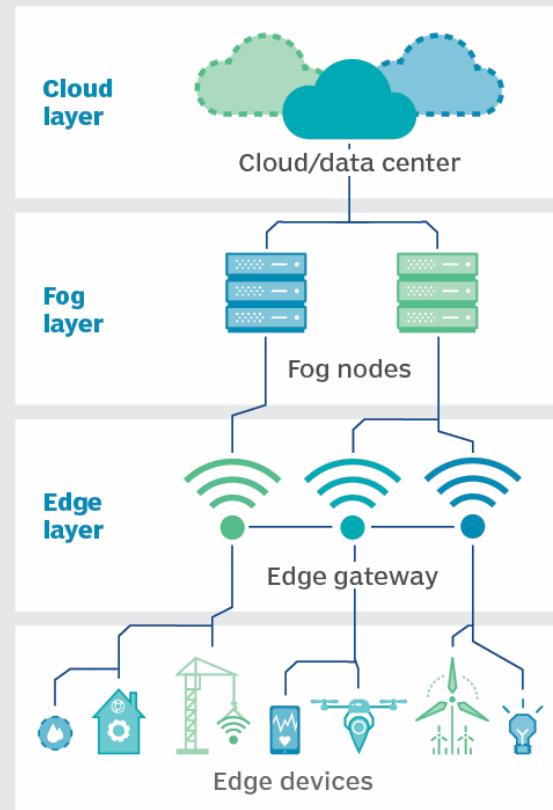


ILLUSTRATION: VECTORSAMA/ADOBE STOCK, VASABII/GETTY IMAGES  
©2020 TECHTARGET. ALL RIGHTS RESERVED

# Advantages of Edge Computing



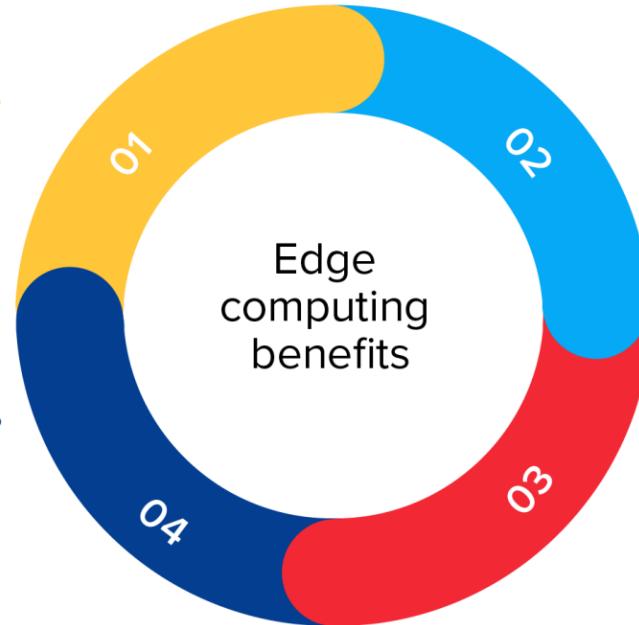
## Latency

Reduction of latency by processing the data closer to the customer



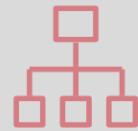
## Security → Privacy

Computing at the edge provides more security than computing at the cloud because it is less vulnerable to numerous variety of threats due to its scope



## Bandwidth

Sending data from edge to the cloud takes up spectral resources; there's just not enough bandwidth for data transportation

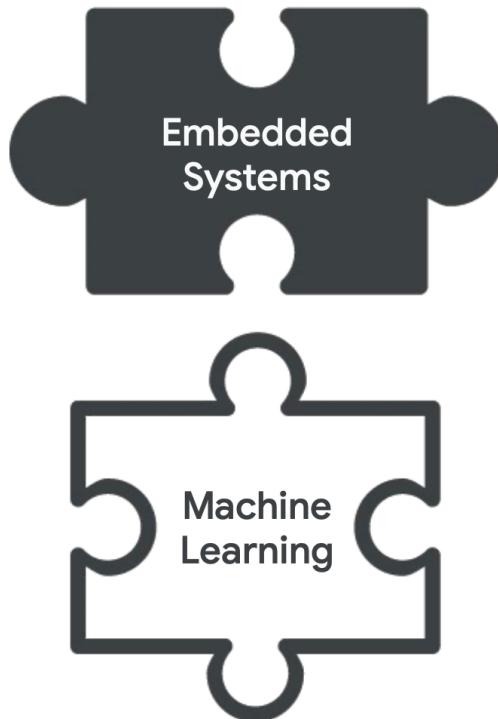


## Reliability

By processing data at the edge, you eliminate network reliability problems

<https://www.wwt.com/article/show-me-the-money-drive-new-revenue-streams-with-edge-computing>

# What is TinyML?



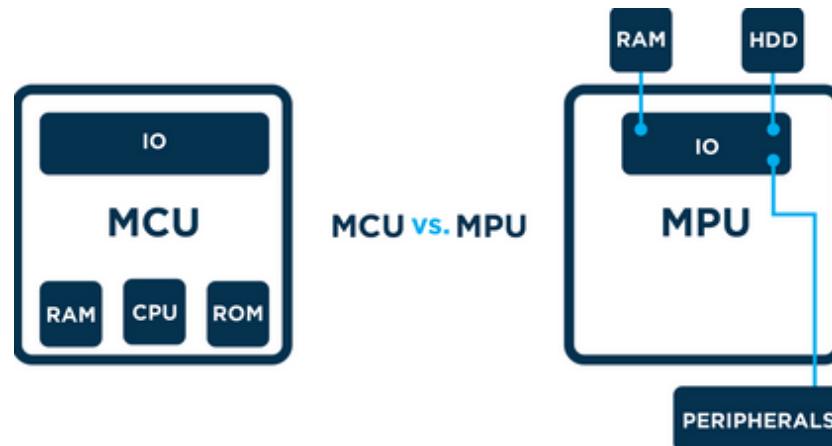
Original definition: "...a neural network model that runs at an energy cost of below 1 mW."  
© "TinyML" by Pete Warden, Daniel Situnayake



## TinyML

"Tiny machine learning (TinyML) is a fast-growing field of machine learning technologies and applications including algorithms, hardware, and software capable of performing on-device sensor data analytics at extremely low power consumption, **typically in the mW range and below**, enabling a variety of always-on ML use-cases **on battery-operated devices**."

- A microcontroller is a microprocessor with built-in memory, timers, and hardware for connecting external devices such as sensors, actuators, and radio transceivers.
- Typically, a smart object microcontroller has a few kilobytes of on-chip memory and is run at a clock speed of a few megahertz.



## Platform

## Compute

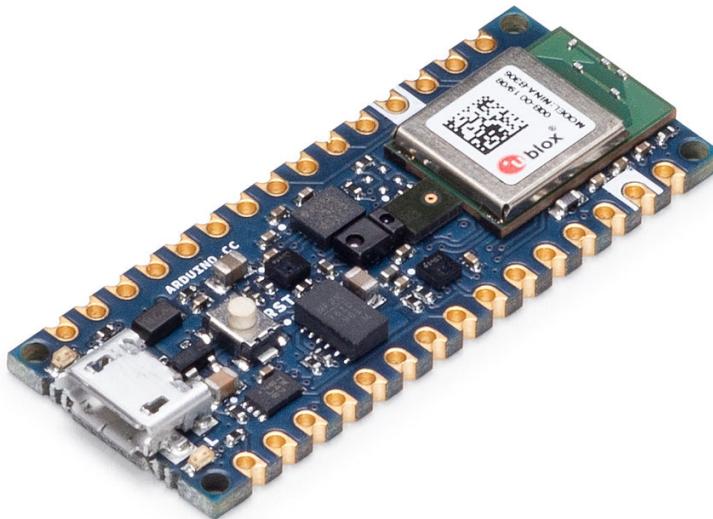
## Memory

## Storage

## Power

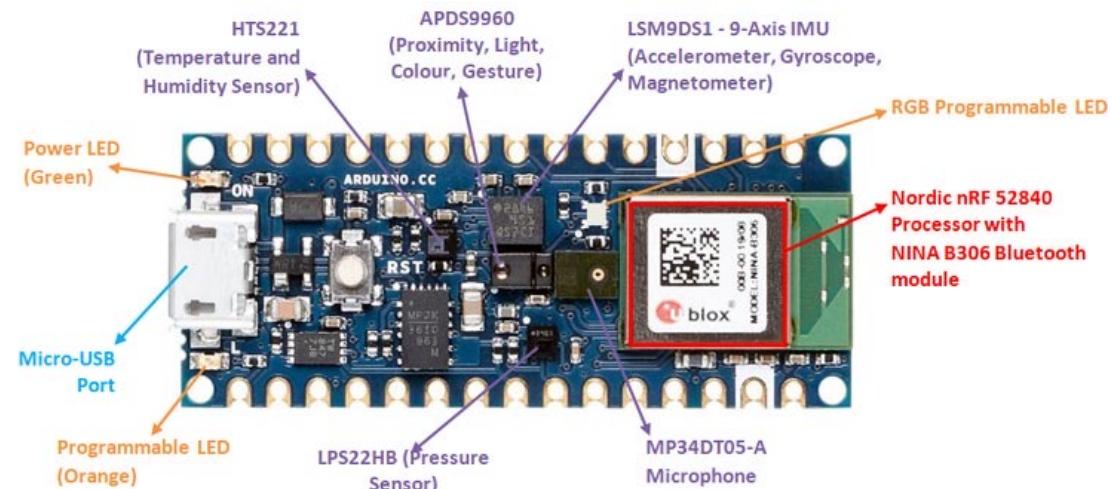
	<b>Microprocessor</b>	<b>&gt;</b>	<b>Microcontroller</b>
Platform			
Compute	1GHz–4GHz	~10X	1MHz–400MHz
Memory	512MB–64GB	~10000X	2KB–512KB
Storage	64GB–4TB	~100000X	32KB–2MB
Power	30W–100W	~1000X	150µW–23.5mW

# Most widely used HW for TinyML (currently)

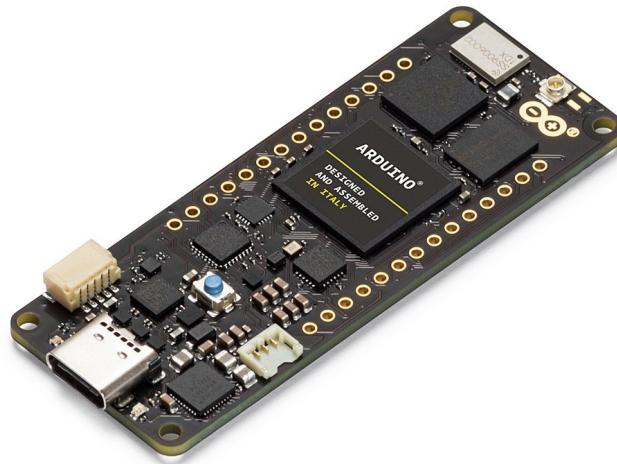


## Arduino Nano 33 BLE Sense

64 MHz Arm® Cortex-M4F (with FPU)  
1 MB Flash + 256 KB RAM  
Bluetooth® 5 multiprotocol radio



# Most widely used HW for TinyML (currently)



**Portenta H7** is probably one of the currently most powerful MCUs.

H7's main processor is the dual core STM32H747 including a Cortex® M7 running at 480 MHz and a Cortex® M4 running at 240 MHz.

The two cores communicate via a *Remote Procedure Call* mechanism.

Both processors share all the in-chip peripherals and can run:

- Arduino sketches on top of the Arm® Mbed™ OS
- Native Mbed™ applications
- MicroPython / JavaScript via an interpreter
- TensorFlow™ Lite

Or even... <https://ouraring.com/>



The future of health wrapped around your finger — monitoring your sleep, heart rate, activity, and temperature with personalized insights.

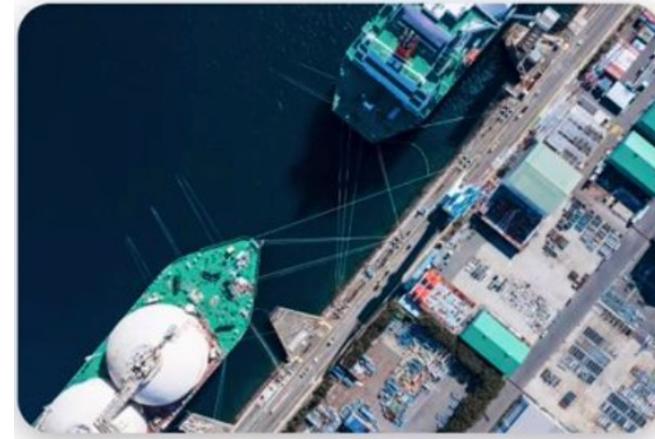
# Applications of TinyML: the beginning



# Applications examples: asset tracking and monitoring

- Locate and manage assets remotely using sensors in real time
- Track health condition through an interactive dashboard
- Measure machine performance and track missing assets

## Asset Tracking & Monitoring



Motion, temp, humidity, position, audio and camera

- Logistics
- Infrastructure
- Buildings

# Applications examples: predictive maintenance

- Using sensors, data collation, analytics and machine learning to monitor equipment continuously and predict failure with far greater accuracy. See for example:  
<https://www.scnsoft.com/blog/iot-predictive-maintenance-guide>
  - **Florida Power & Light** has turned to IoT development to deploy a predictive maintenance solution that estimates when turbines are running ineffectively or about to fail
  - **Chevron** has turned to IoT development to roll out a predictive maintenance solution that helps to identify corrosion and pipeline damages.

RAM-1: An Advanced Grid Monitoring Solution

<https://www.ram-center.com>



# But the range of applications is wide...

We help African smallholder farmers adapt to climate change

Help us reach more farmers

[Donate ❤ & support farmers →](#)



Plant Village: <https://plantvillage.psu.edu>

# But the range of applications is wide...



The most impactful way to stop climate change? Save rainforests.

## HOW OUR SYSTEM HELPS PRESERVE RAINFORESTS



### PREVENT ILLEGAL DEFORESTATION

Our mission is to enable our partners-on-the-ground to protect rainforests. Our system sends real-time alerts for chainsaws, trucks, cars and signs of incursion. [Read more](#)



### HALT ANIMAL POACHING

We can help stop poaching by providing our partners with real-time data and patterns of activity that allow for targeted protections in key strategic areas. [Read more](#)



### ENABLE BIO-ACOUSTIC MONITORING

There's more to a rainforest than meets the eye. Our bio-acoustic monitoring capabilities give partners new ways of understanding rainforests. [Read more](#)

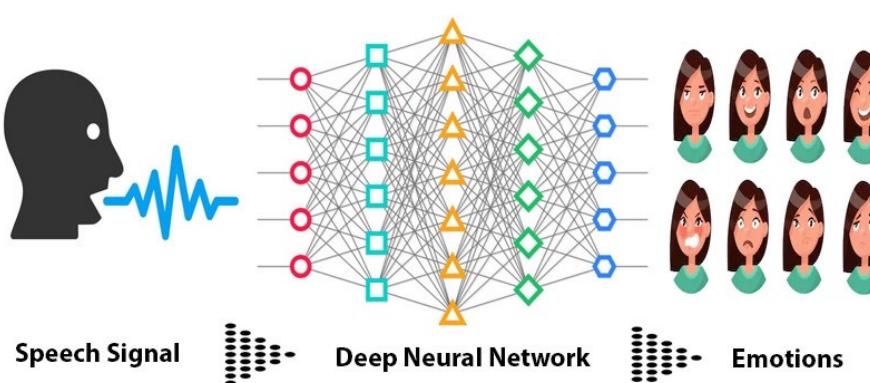
Rainforest Connection: <https://rfcx.org/ecoacoustics>

And...

## ~~People counting~~



## Sentiment analysis



### TensorFlow Lite example apps

Explore pre-trained TensorFlow Lite models and learn how to use them in sample apps for a variety of ML applications.

<p><b>Image classification</b> Identify hundreds of objects, including people, activities, animals, plants, and places. <b>Model overview</b> → Try it on Android ⓘ Try it on iOS ⓘ Try it on Raspberry Pi ⓘ</p>	<p><b>Object detection</b> Detect multiple objects with bounding boxes. Yes, dogs and cats too. <b>Model overview</b> → Try it on Android ⓘ Try it on iOS ⓘ Try it on Raspberry Pi ⓘ</p>	<p><b>Pose estimation</b> Estimate poses for single or multiple people. Imagine the possibilities, including stick figure dance parties. <b>Model overview</b> → Try it on Android ⓘ Try it on iOS ⓘ Try it on Raspberry Pi ⓘ</p>
<p><b>Speech recognition</b> Identify speech commands by recognizing keywords.  Try it on Android ⓘ Try it on iOS ⓘ</p>	<p><b>Gesture recognition</b> Recognize gestures using your webcam.  Try it on Android ⓘ Try it on iOS ⓘ</p>	<p><b>Segmentation</b> Pinpoint the shape of objects with strict localization accuracy and semantic labels. Trained with people, places, animals, and more. <b>Model overview</b> → Try it on Android ⓘ Try it on iOS ⓘ Try it on Raspberry Pi ⓘ</p>
<p><b>Text classification</b> Categorize free text into predefined groups. Potential applications include abusive content moderation, tone detection, and more. <b>Model overview</b> → Try it on Android ⓘ</p>	<p><b>On-device recommendation</b> Provide personalized on-device recommendations based on events selected by users. <b>Model overview</b> → Try it on Android ⓘ</p>	<p><b>Natural language question answering</b> Answer questions based on the content of a given passage of text with BERT. <b>Model overview</b> → Try it on Android ⓘ Try it on iOS ⓘ</p>

<https://www.tensorflow.org/lite/examples>



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA

# Current limitations



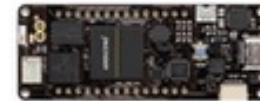
Anomaly Detection  
Sensor Classification  
20 KB



Rpi-Pico  
(Cortex-M0+)



Arduino Nano  
(Cortex-M4)



Arduino Pro  
(Cortex-M7)



Raspberry Pi  
(Cortex-A)

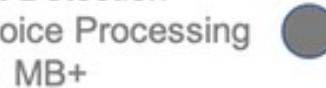


Jetson Nano  
(Cortex-A + GPU)

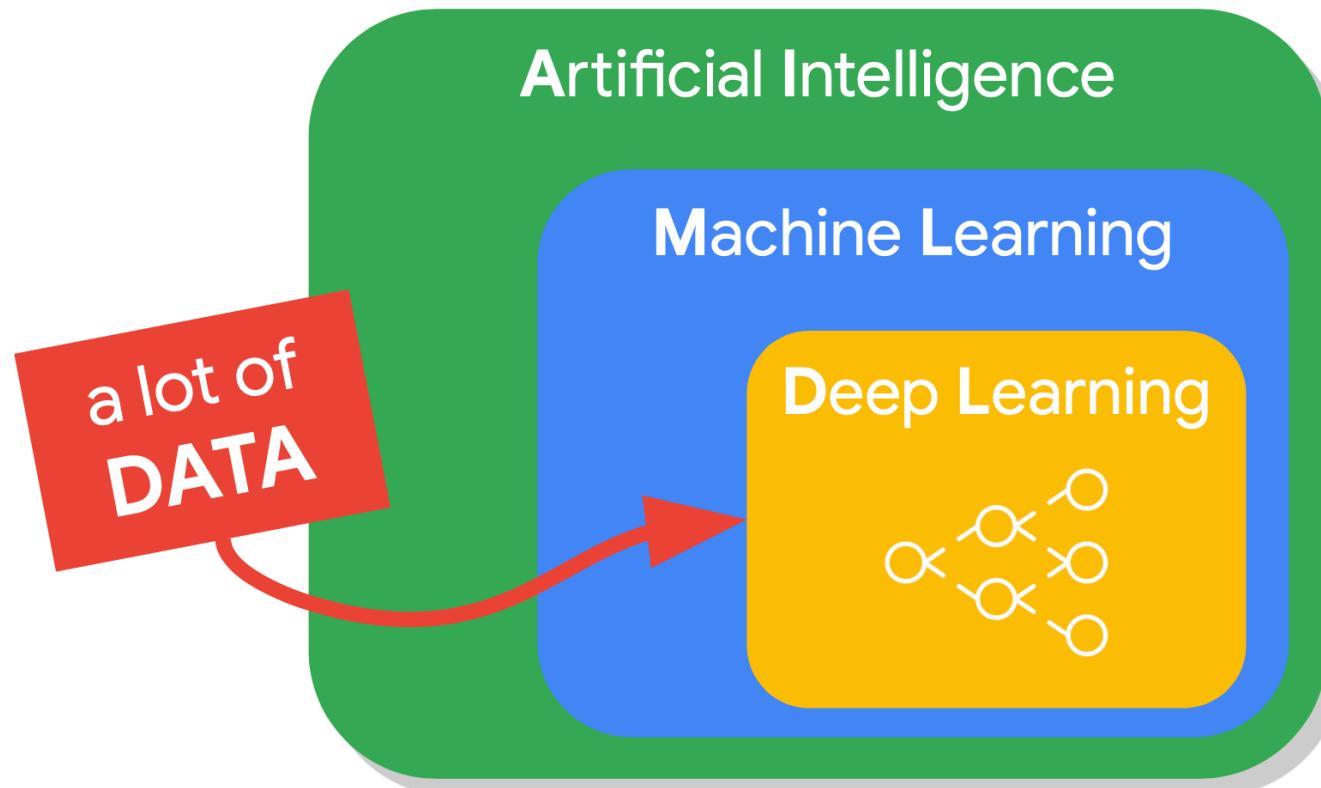
Image Classification  
250 KB+

Object Detection  
Complex Voice Processing  
1 MB+

Video Classification  
2 MB+

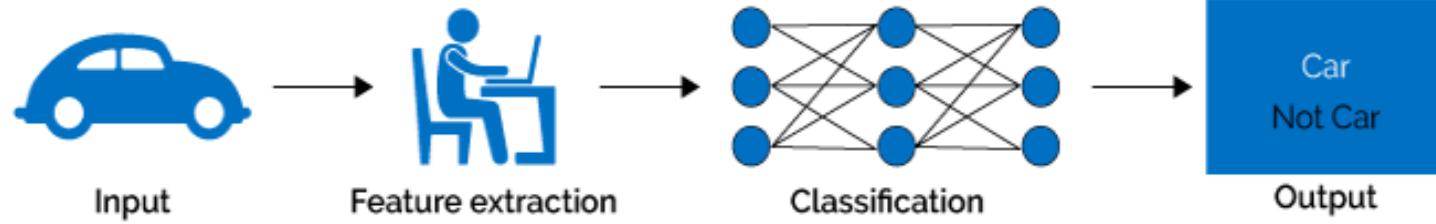


# Machine Learning: Some initial definitions

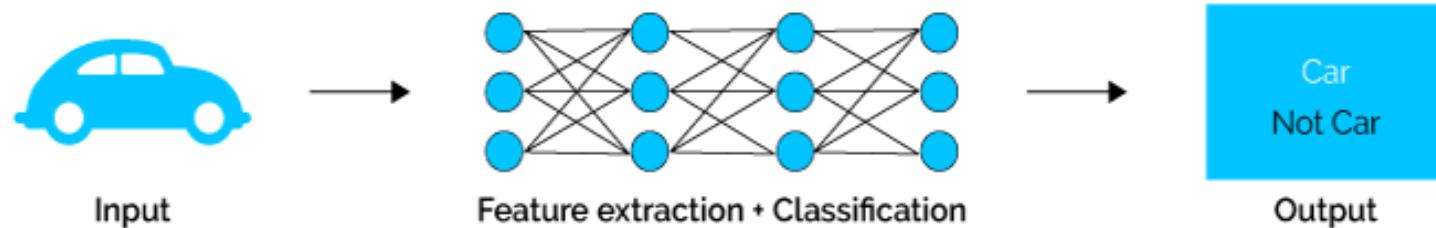


# Machine Learning vs Deep Learning

## Machine Learning



## Deep Learning



## Machine Learning



## Training the machine

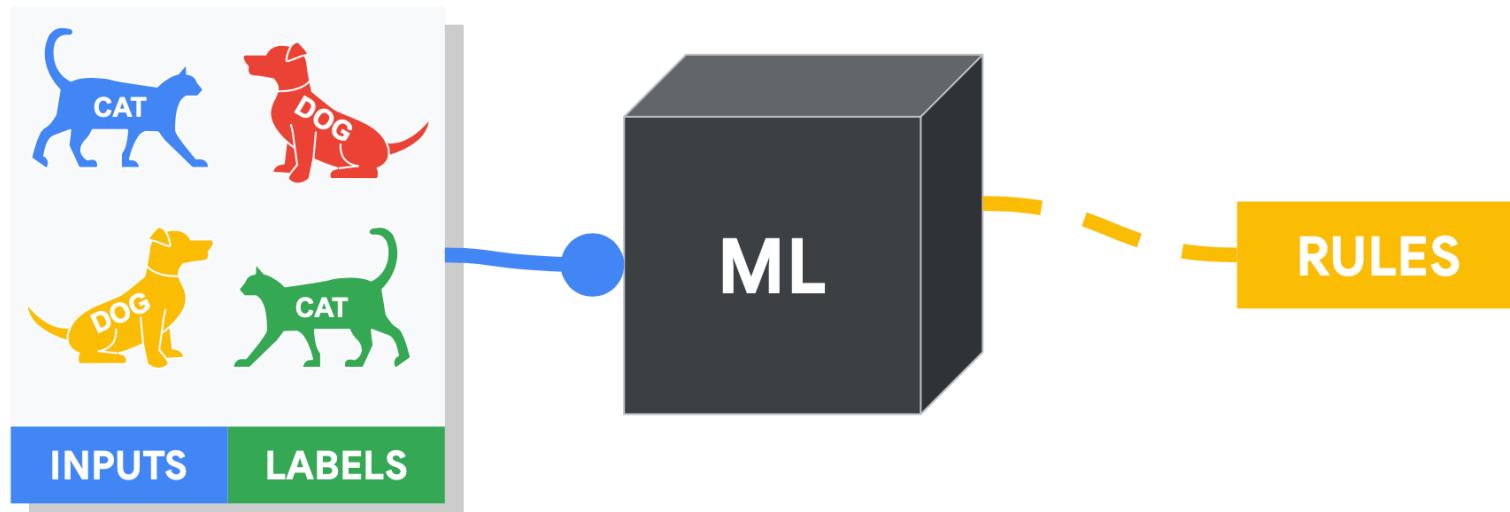


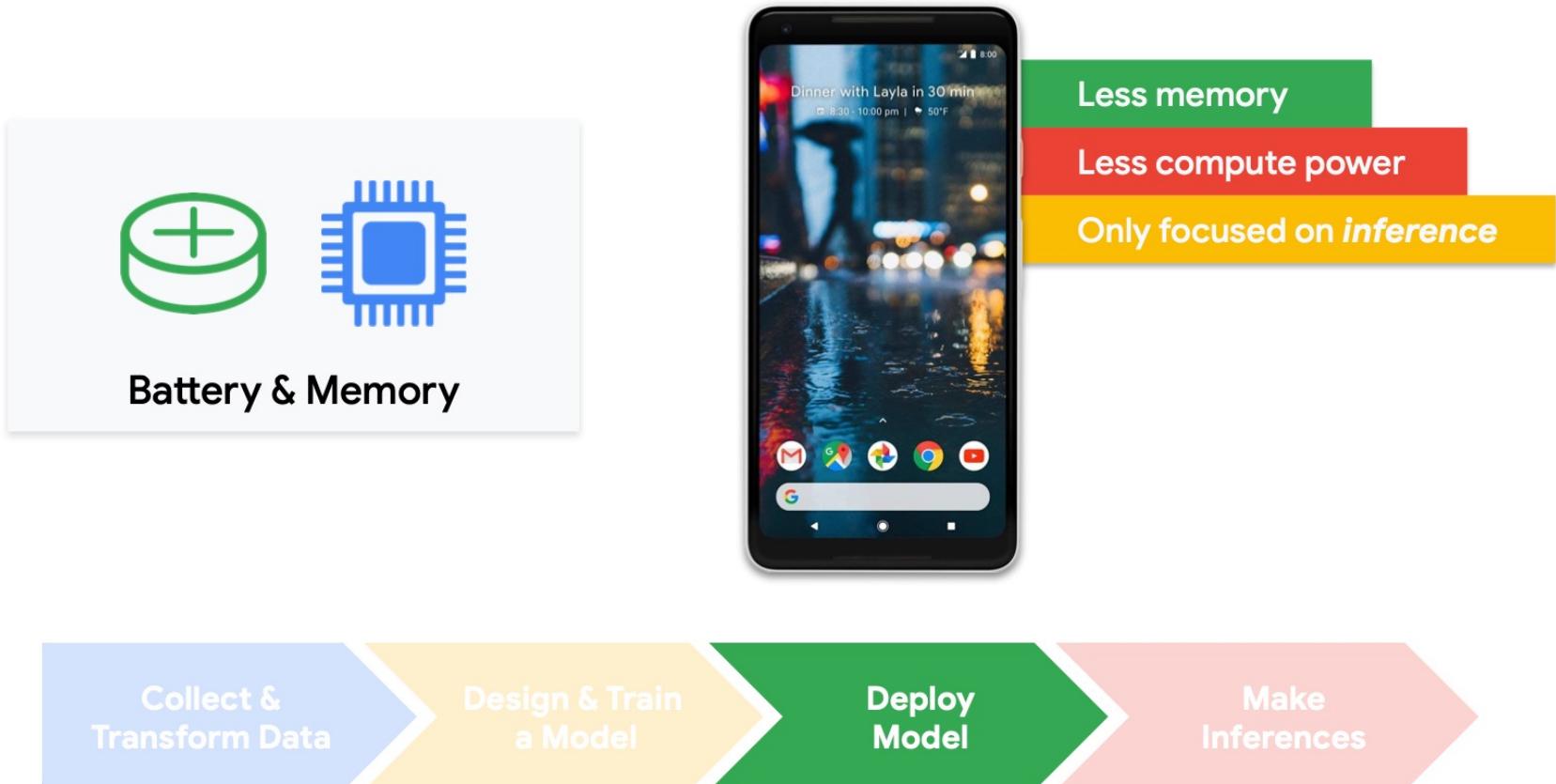
WE PROVIDE

LABELS

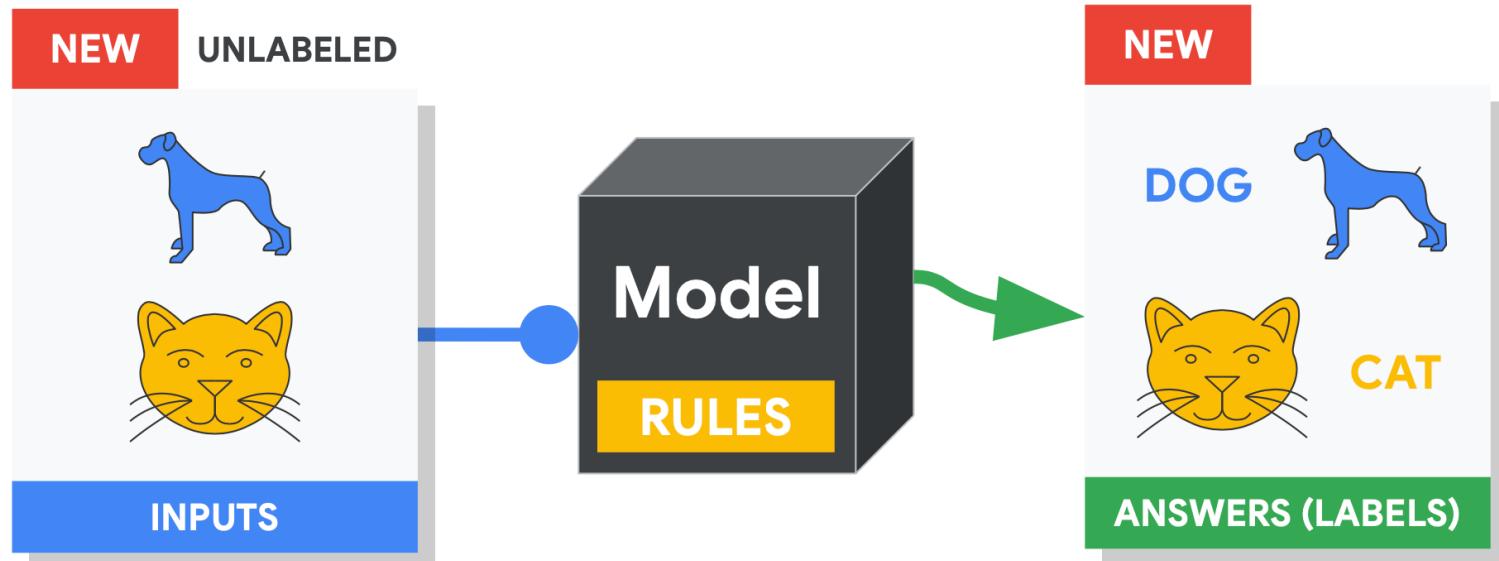


## Training the machine



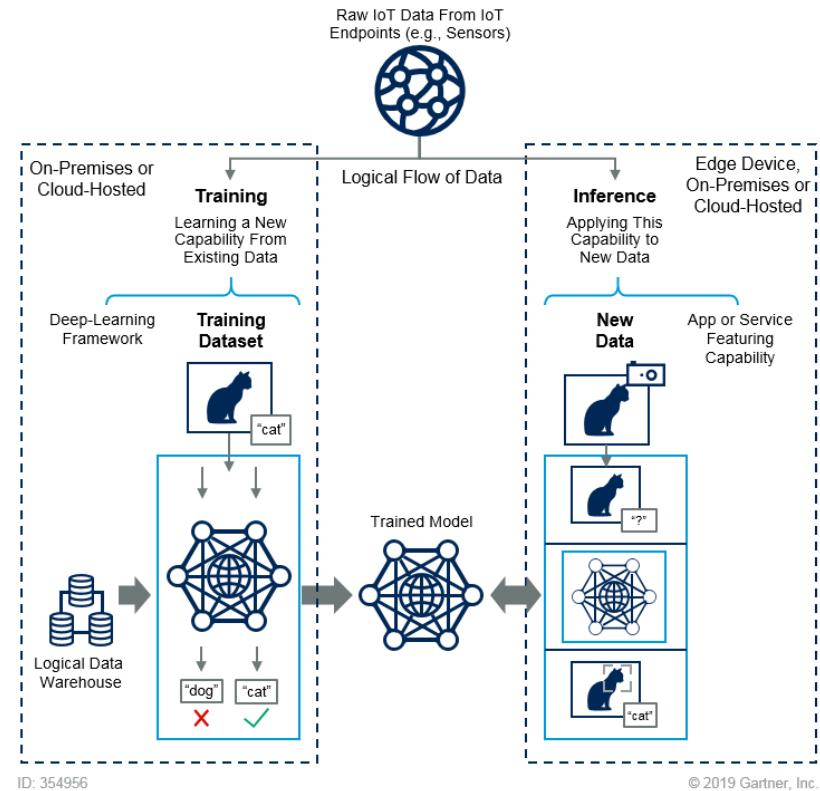


## Making predictions:



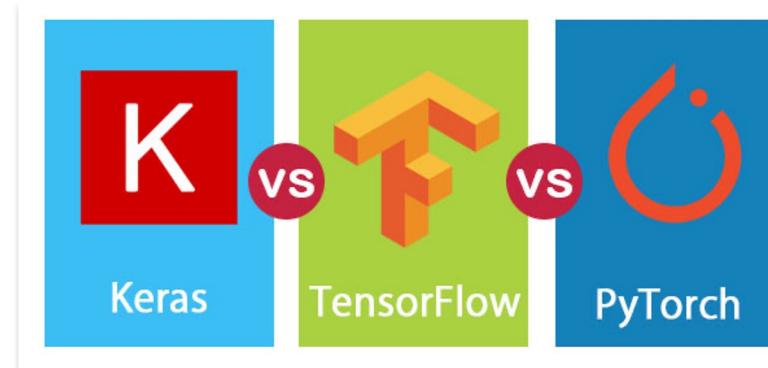
# Training and Inference

- Machine Learning occurs in two stages – learning and inferencing. At present, TinyML only handles inferencing
- Training refers to the process of creating a machine learning algorithm. Training involves using a machine learning framework and a training dataset.
  - IoT data provides a source of training data that data scientists and engineers can use to train machine learning models.
- Inference refers to the process of using a trained machine-learning algorithm to make a prediction.
  - IoT data can be used as the input to a trained machine learning model, enabling predictions that can guide decision logic on the device, at the edge.



# About the available software frameworks

- The three main frameworks which are available as an open-source library are opted by data scientist in deep learning are TensorFlow, Keras, and PyTorch.
  - **TensorFlow** is used to perform multiple tasks in data flow programming and machine learning applications.
  - **Keras** is a neural network library scripted in Python and executes on the top layer of TensorFlow.
  - **PyTorch** is a machine learning library that is mainly used in natural language processing.



# TensorFlow Lite for microcontrollers

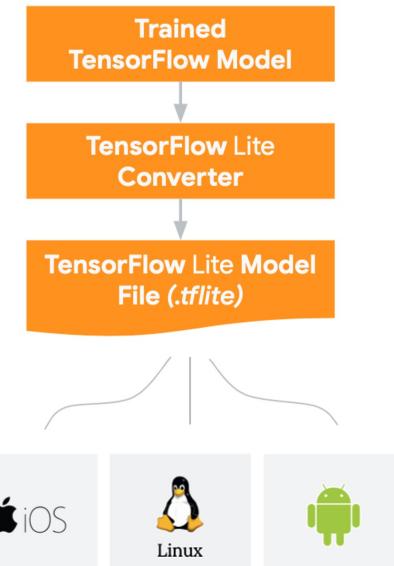
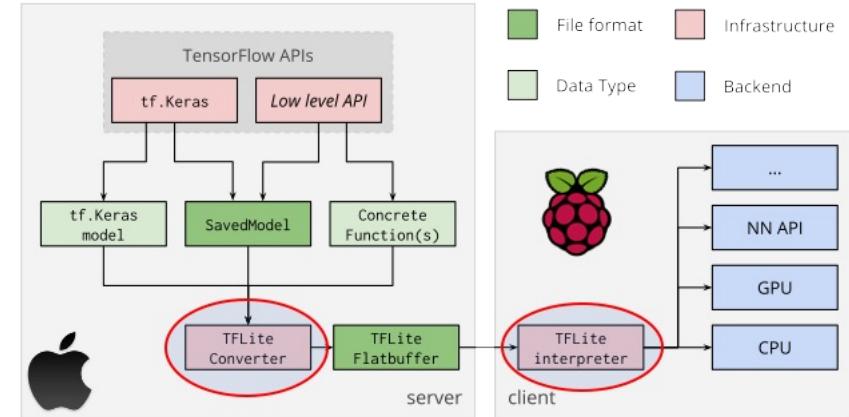


TensorFlow Lite

- <https://www.tensorflow.org/lite/microcontrollers>
- TensorFlow is Google's open-source machine learning framework which helps in developing machine learning models quickly.
- For TinyML, there is TensorFlow Lite Micro, which is a specialised version of TensorFlow for microcontrollers.
  - TFLite Micro is written in C++ 11 and requires a 32-bit platform and is mostly compatible with Arm Cortex-M Series processors.
  - [An ESP32 port is available as well](#)
- TFLite Micro allows you to easily compress regular TensorFlow models into just a few kilobytes and comes with numerous example models.

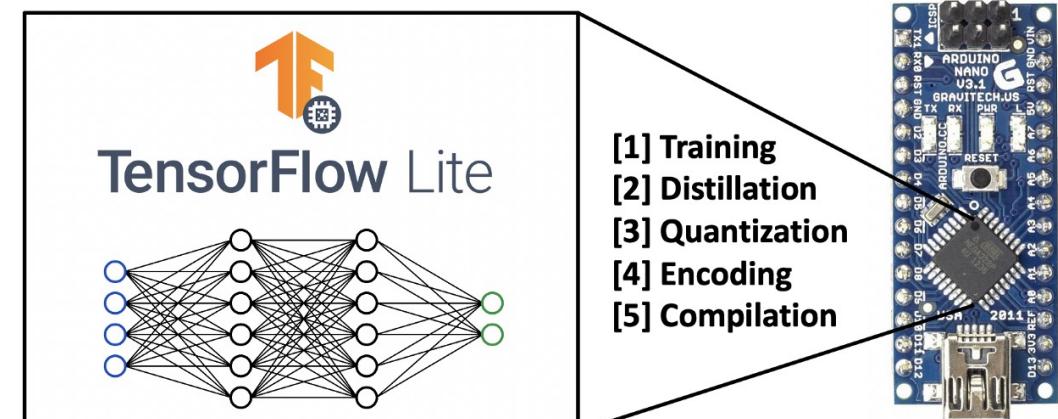
# TensorFlow Lite (TFLite)

- Consists of two main components:
  - The **TFLite converter**, which converts TensorFlow models into an efficient form for use by the interpreter and can introduce optimizations to improve binary size and performance.
  - The **TFLite interpreter** runs with specially optimized models on many different hardware types, including mobile phones, embedded Linux devices, and microcontrollers.



# TensorFlow Lite optimizations

- Problem:
  - Less Compute Power, Less Memory, High Accuracy
- Some optimizations:
  - Quantization
  - Pruning
- ..and
  - Separable Convolutions
  - Knowledge Distillation

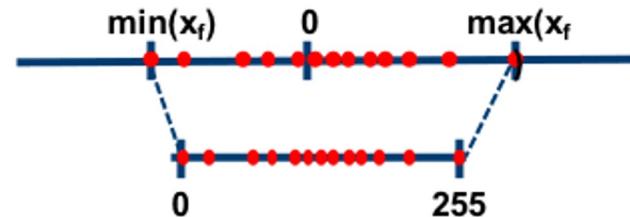
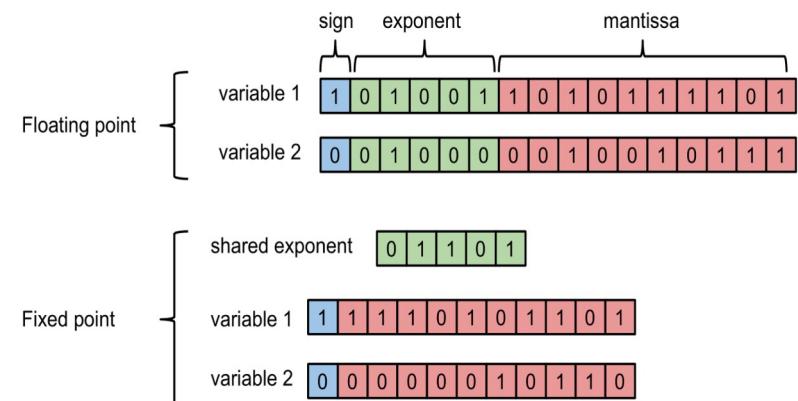


## TinyML



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA

- Neural Networks generally use 32 bit floating point weights
- These require more memory to save and custom logic to execute efficiently (which many micro-controllers do not have)
- By converting weights to INT8 (8 bits), we can save memory and compute faster
- However, it leads to a loss in accuracy since we lose precision



[source: heartbeat.fritz.ai](http://heartbeat.fritz.ai)

# TensorFlow Lite on the Raspberry Pi

- The example below shows how to use TensorFlow Lite on the Raspberry Pi to run object detection models.

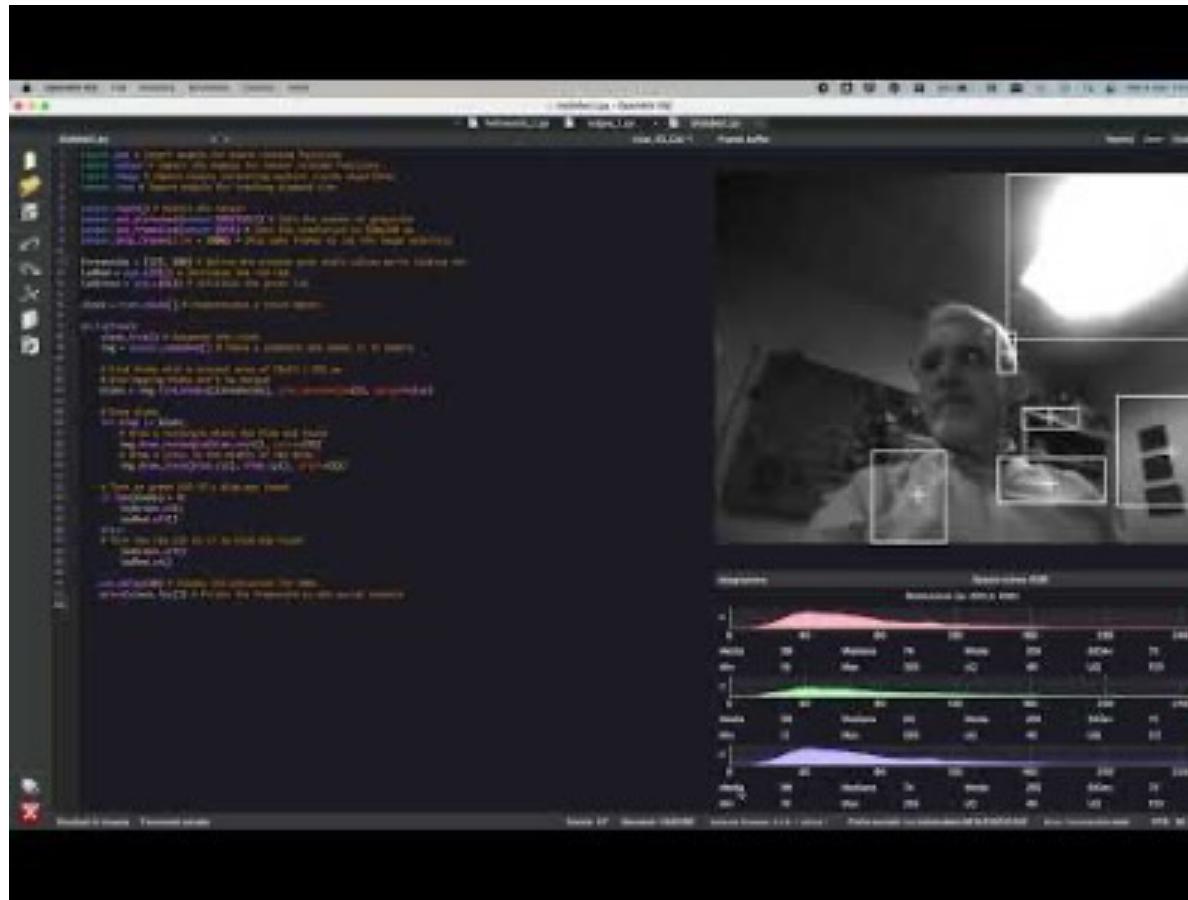


<https://www.youtube.com/watch?v=b7Yul7NxNq0>

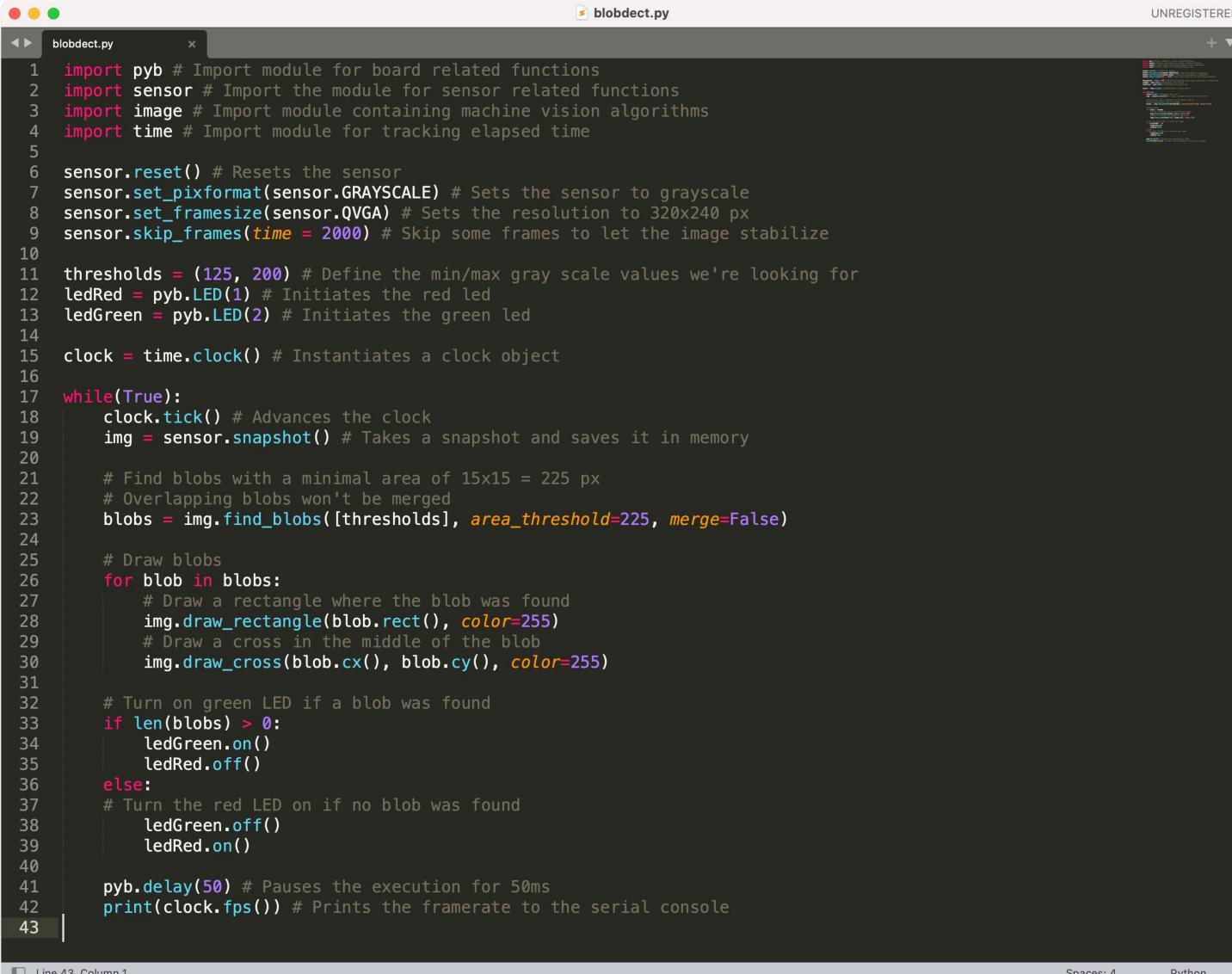
- <http://openmv.io/>
- OpenMV is another TinyML development platform, this time specialising in computer vision applications. This includes machine learning applied onto any kind of image or video like object recognition or image classification.
- The platform is built around their OpenMV Cam H7, which is a microcontroller board that runs on the ARM Cortex M7 processor and is Micropython programmable.
- In addition, OpenMV offers a cross-platform IDE that features a powerful code editor, debug terminal, and framebuffer viewer with histogram display – all of which are key components of developing computer vision applications in TinyML!

# OpenMV: Blob detection with Portenta

- This example performs “Blob Detection” with a Portenta detecting the presence and the position of objects in a camera image.



# OpenMV: Blob detection with Portenta: the code



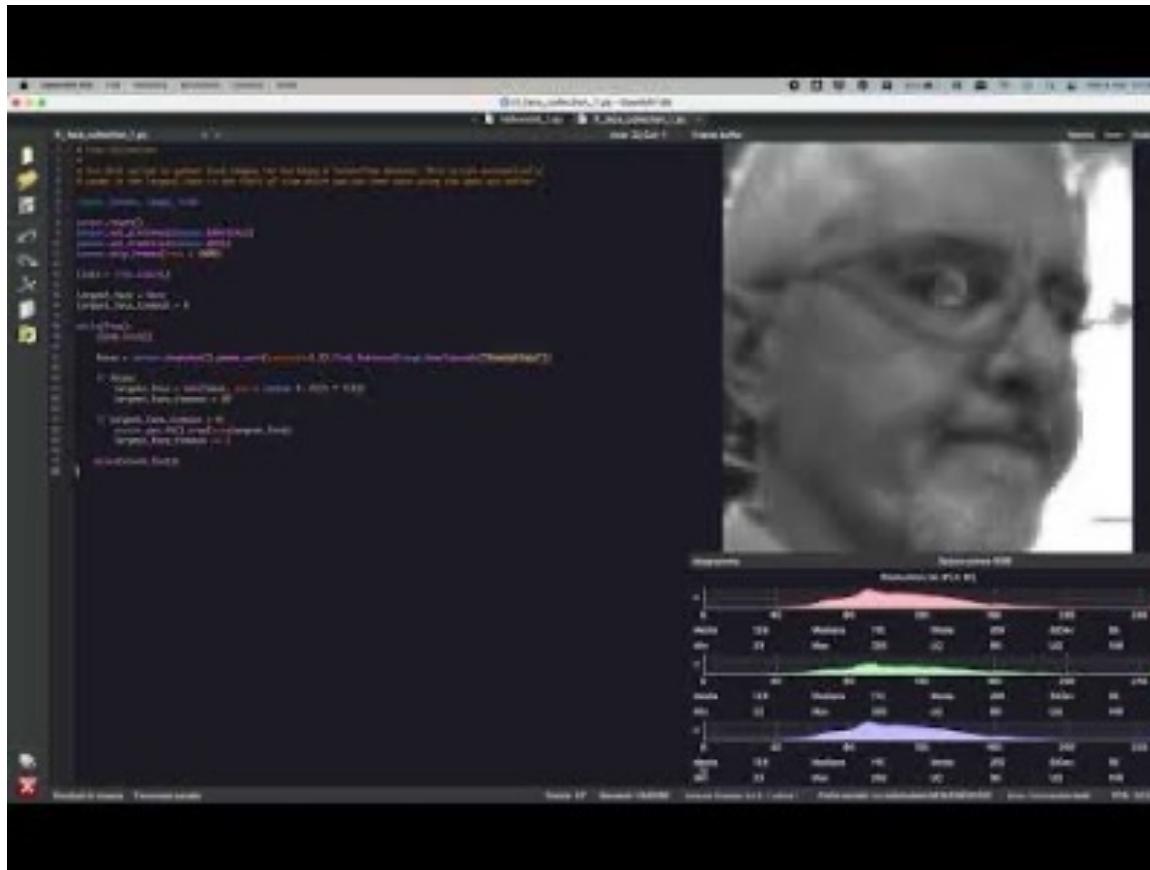
The screenshot shows a terminal window titled "blobdetect.py" with the following Python code:

```
blobdetect.py x UNREGISTERED
1 import pyb # Import module for board related functions
2 import sensor # Import the module for sensor related functions
3 import image # Import module containing machine vision algorithms
4 import time # Import module for tracking elapsed time
5
6 sensor.reset() # Resets the sensor
7 sensor.set_pixformat(sensor.GRAYSCALE) # Sets the sensor to grayscale
8 sensor.set_framesize(sensor.QVGA) # Sets the resolution to 320x240 px
9 sensor.skip_frames(time = 2000) # Skip some frames to let the image stabilize
10
11 thresholds = (125, 200) # Define the min/max gray scale values we're looking for
12 ledRed = pyb.LED(1) # Initiates the red led
13 ledGreen = pyb.LED(2) # Initiates the green led
14
15 clock = time.clock() # Instantiates a clock object
16
17 while(True):
18     clock.tick() # Advances the clock
19     img = sensor.snapshot() # Takes a snapshot and saves it in memory
20
21     # Find blobs with a minimal area of 15x15 = 225 px
22     # Overlapping blobs won't be merged
23     blobs = img.find_blobs([thresholds], area_threshold=225, merge=False)
24
25     # Draw blobs
26     for blob in blobs:
27         # Draw a rectangle where the blob was found
28         img.draw_rectangle(blob.rect(), color=255)
29         # Draw a cross in the middle of the blob
30         img.draw_cross(blob.cx(), blob.cy(), color=255)
31
32     # Turn on green LED if a blob was found
33     if len(blobs) > 0:
34         ledGreen.on()
35         ledRed.off()
36     else:
37         # Turn the red LED on if no blob was found
38         ledGreen.off()
39         ledRed.on()
40
41     pyb.delay(50) # Pauses the execution for 50ms
42     print(clock.fps()) # Prints the framerate to the serial console
43
```

Line 43, Column 1 | Spaces: 4 | Python

# OpenMV: Face detection with Portenta

- This other example performs recognizes faces



`tf_face_collection_1.py`

# OpenMV: Face detection with Portenta: the code

The screenshot shows a Python code editor window titled "tf\_face\_collection\_1.py". The code is a script for collecting face images using a sensor connected to an OpenMV Portenta board. The script uses TensorFlow's Haar Cascade algorithm to detect faces in the video feed from the sensor. It includes code for initializing the sensor, setting its parameters (RGB565 format, QVGA frame size), and skipping frames. A while loop continuously takes snapshots, applies gamma correction, finds features using the HaarCascade("frontalface") classifier, and identifies the largest face. If a face is detected, it is cropped from the frame. The script also prints the current frame rate (fps). The code editor interface includes tabs for "tf\_face\_collection\_1.py" and "UNREGISTERED", and various status indicators at the bottom.

```
tf_face_collection_1.py x tf_face_collection_1.py UNREGISTERED
1 # Face Collection
2 #
3 # Use this script to gather face images for building a TensorFlow dataset. This script automatically
4 # zooms in the largest face in the field of view which you can then save using the data set editor.
5
6 import sensor, image, time
7
8 sensor.reset()
9 sensor.set_pixformat(sensor.RGB565)
10 sensor.set_framesize(sensor.QVGA)
11 sensor.skip_frames(time = 2000)
12
13 clock = time.clock()
14
15 largest_face = None
16 largest_face_timeout = 0
17
18 while(True):
19     clock.tick()
20
21     faces = sensor.snapshot().gamma_corr(contrast=1.5).find_features(image.HaarCascade("frontalface"))
22
23     if faces:
24         largest_face = max(faces, key = lambda f: f[2] * f[3])
25         largest_face_timeout = 20
26
27     if largest_face_timeout > 0:
28         sensor.get_fb().crop(roi=largest_face)
29         largest_face_timeout -= 1
30
31     print(clock.fps())
32
```

Line 30, Column 1 Spaces: 4 Python

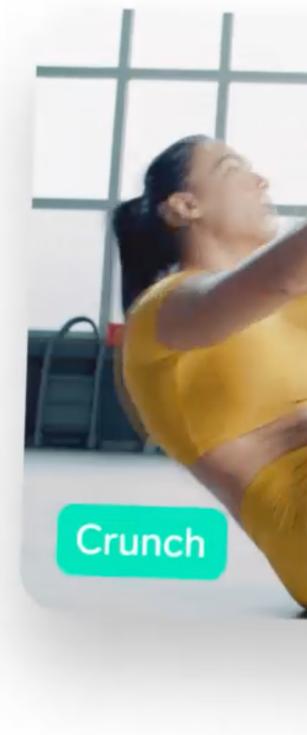
- <http://edgeimpulse.com>
- Edge Impulse is a platform that specifically targets the development of TinyML applications. With an easy-to-use web-based interface, Edge Impulse is arguably the easiest solution for anyone to collect data, train a model, and finally deploy it on a microcontroller. Best of all, it's free for developers, albeit with some limits on the number of projects you can create and compute time.
- Developing TinyML with Edge Impulse also allows you to take advantage of their Edge Optimised Neural (EON) compiler, which can run a neural network with 25-55% less RAM and 35% less storage compared to TFLite for Microcontrollers – a significant point to consider!
- Like TFLite Micro, Edge Impulse supports a substantial list of microcontrollers and development boards, also including the Arduino Nano 33 BLE Sense and Wio Terminal. Supported devices can easily record and upload datasets in a matter of minutes, but other devices can also use their Data forwarder to do the same with just a little more effort.

An alternative: <https://www.lobe.ai>

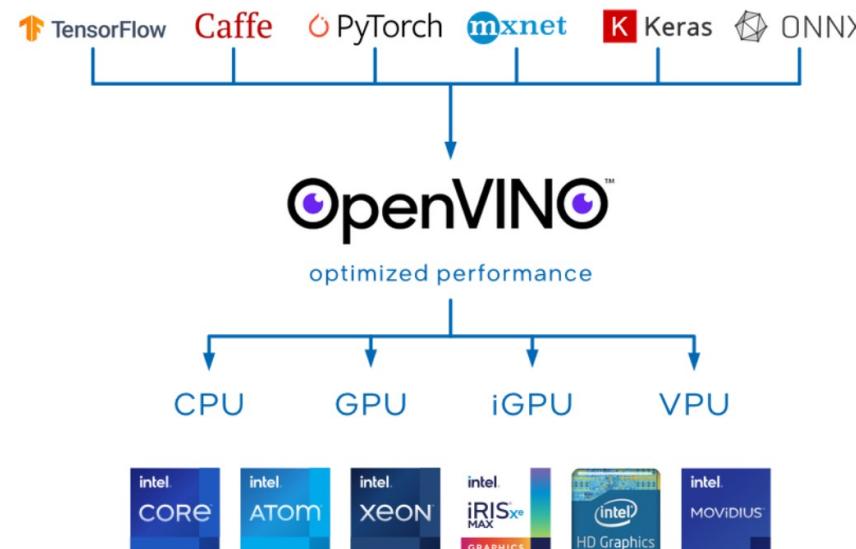
[Overview](#)[Examples](#)[Tour](#)[Blog](#)[Help](#)

# Train apps to count reps

Lobe helps you train machine learning models with a free, easy to use tool.

[Download](#)[Watch Tour](#)   
**Crunch**

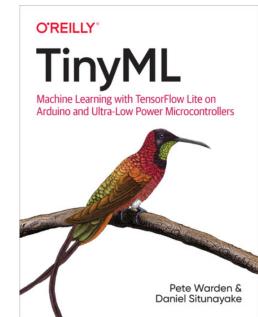
- OpenVINO™ is an open-source toolkit for optimizing and deploying AI inference.
  - Boost deep learning performance in computer vision, automatic speech recognition, natural language processing and other common tasks
  - Use models trained with popular frameworks like TensorFlow, PyTorch and more
  - Reduce resource demands and efficiently deploy on a range of Intel® platforms from edge to cloud



<https://docs.openvino.ai>

# On-line references

- Main refs
  - <https://learning.oreilly.com/library/view/tinyml/9781492052036/>
  - [video Pete Warden: Building with TensorFlow Lite for microcontrollers | Workshop](#)
- Tiny Machine Learning Community
  - <https://discuss.tinymlx.org>
- HarvardX Profession Certificate in Tiny Machine Learning (TinyML)
  - <https://github.com/tinyMLx/courseware>
- Tensorflow lite:
  - <https://www.tensorflow.org/lite/>
  - <https://experiments.withgoogle.com/collection/tfliteformicrocontrollers>
- [Seeedstudio: Everything About TinyML – Basics, Courses, Projects & More!](#)



# TinyML: Machine Learning meets the Internet of Things



Thanks to:

- **Marcelo Rovai**
  - <https://github.com/Mjrovai/UNIFEI-IESTIoT-TinyML-2021.2>
- **Vijay Janapa Reddi**
  - <https://tinyml.seas.harvard.edu/SciTinyML/schedule/>
- Archana Vaidheeswaran
- Soham Chatterjee
- ...and many more!
  - <https://tinyml.seas.harvard.edu/CRESTLEX3/>