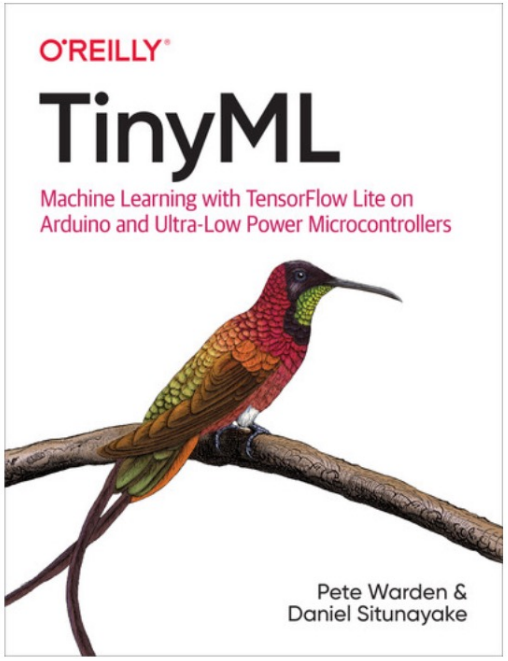
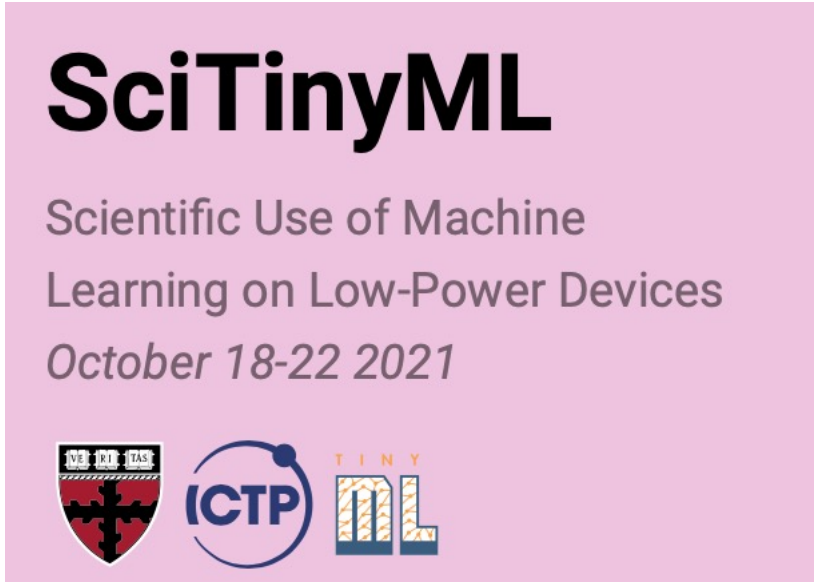
TinyML: Machine Learning meets the Internet of Things



https://learning.oreilly.com/library/view/tinyml/9781492052036/

This presentation is strongly based on:



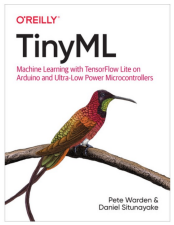
https://tinyml.seas.harvard.edu/SciTinyML/schedule/

Thanks to:

- **Marcelo Rovai**
- **Vijay Janapa Reddi**
- Archana Vaidheeswaran
- Soham Chatterjee
- ...and many more!
  - https://tinyml.seas.harvard.edu/CRESTLEX3/

https://github.com/Mjrovai/UNIFEI-IESTI01-TinyML-2021.2

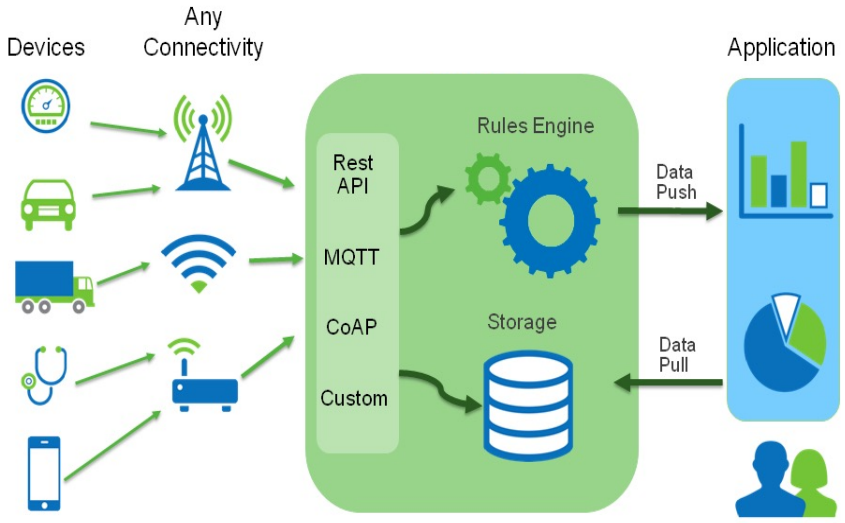
On-line references



- 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!

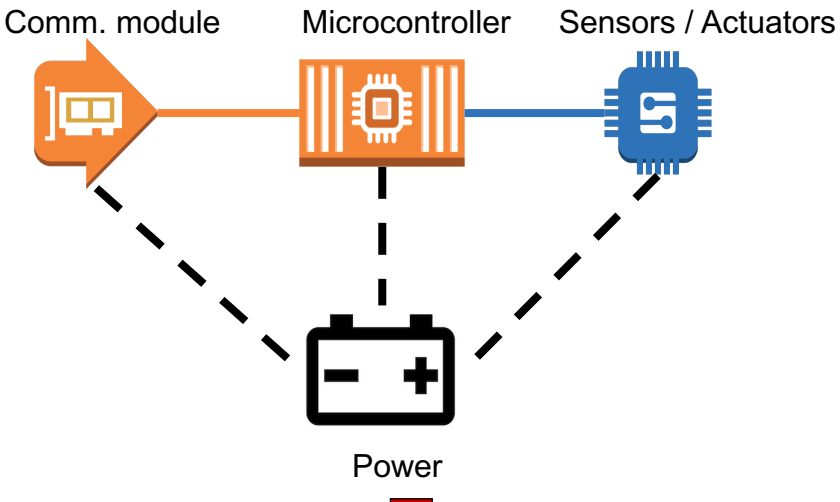
IoT simplified model

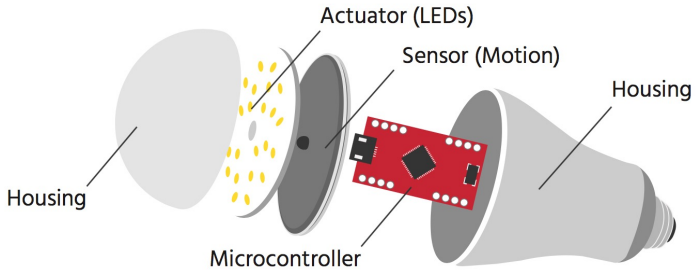
- Devices ("things")
  - These could be sensors, actuators, robots, cars, whatever can be connected.
  - A lot of inheritance from the world of "sensors networks"
- Connectivity
  - To connect things reliably to Internet.
  - Wireless connectivity is central to this task
- Platform
  - the collected data needs to be stored and processed somewhere. Typically cloud-based infrastructures



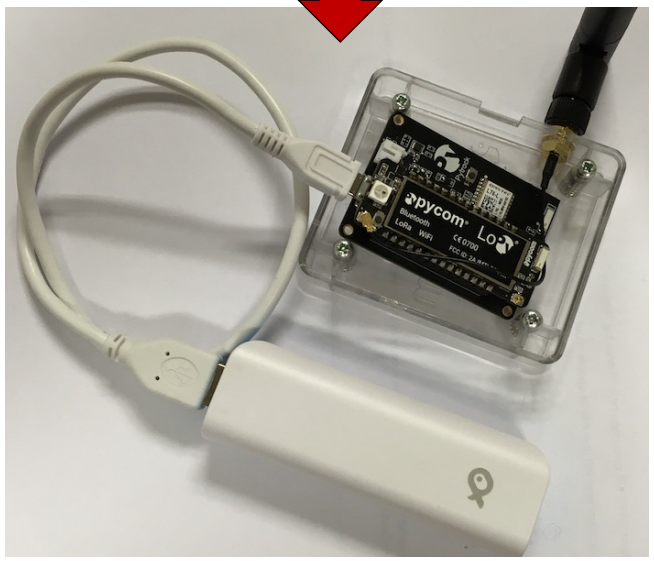
Things

- 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.



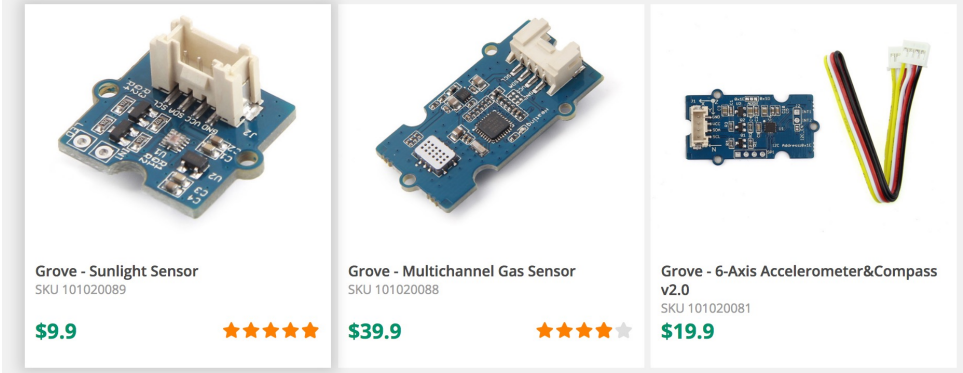


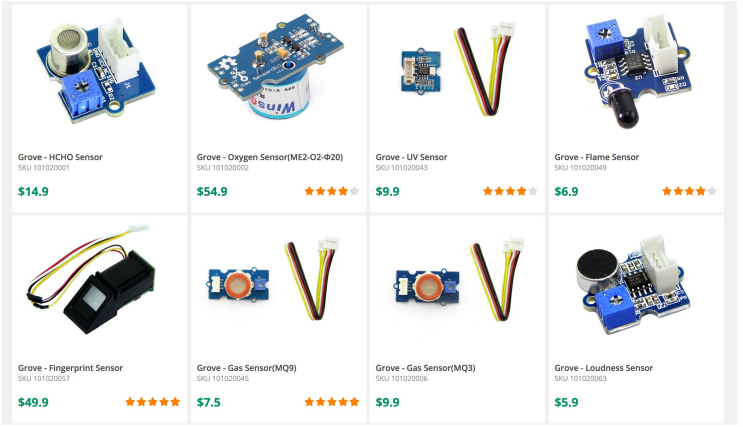
A Reference Guide to the Internet of Things Copyright © 2017 Bridgera LLC, RIoT



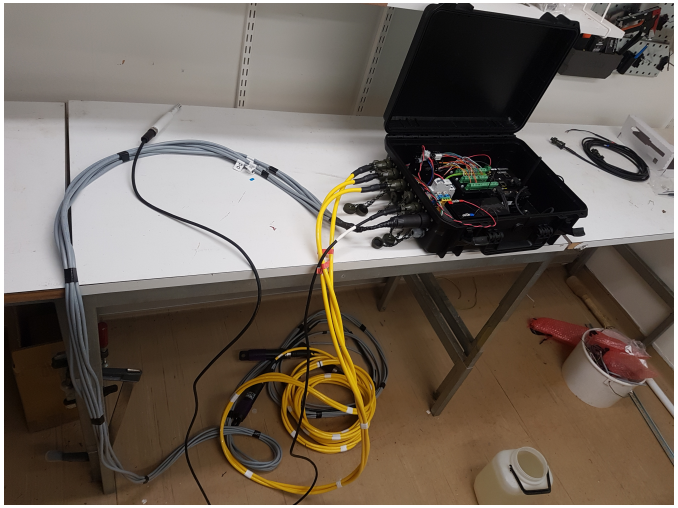
Classic Sensors

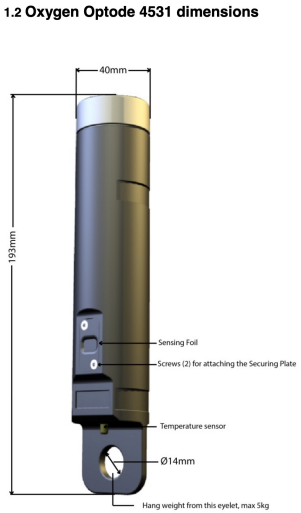
cheap…





expensive…



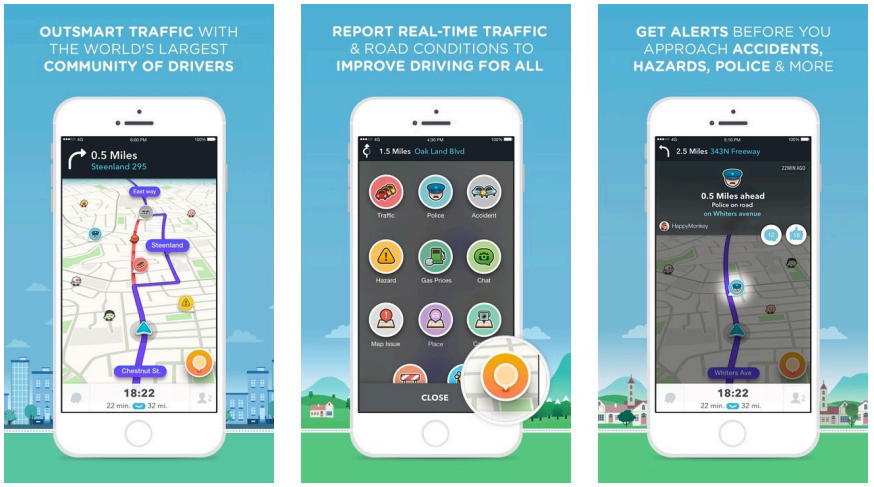


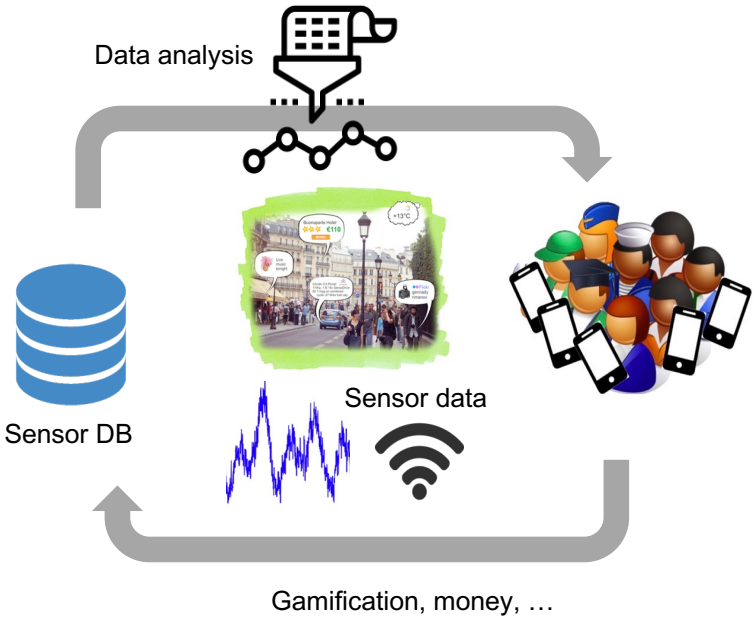
Parameter	Output	Default range[2)]	Calibrated range	Accuracy	Resolution
Oxygen Concentration	0 - 5V	0 to 800µM	0 to 500µM	<8µM or 5% whichever is greater	< 1µM
	4 - 20mA	0 to 800µM	0 to 500µM	<9µM or 5.2% whichever is greater	< 1µM
Oxygen Saturation	0 - 5V	0 – 200%	0 - 120%	<5 %	<0.4%
	4 - 20mA	0 – 200%	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

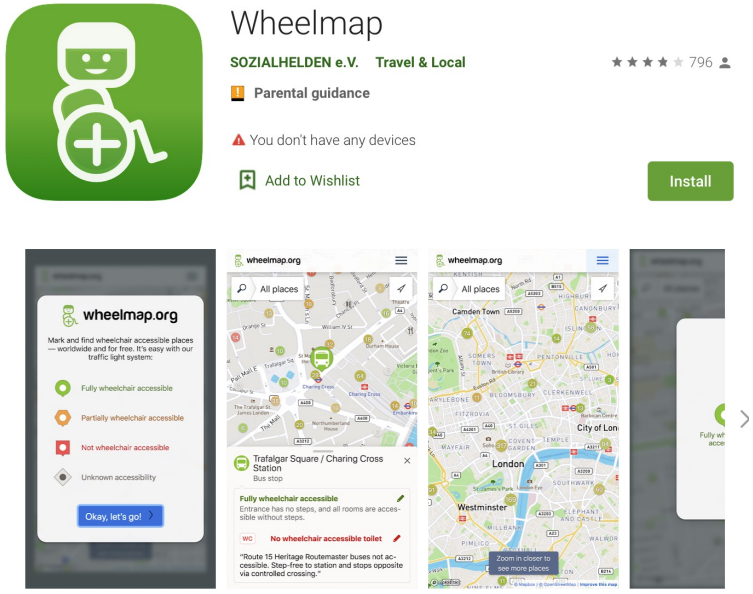
Beyond conventional sensors

- Humans as a sensor
  - social sensors: E.g., tweeting real-world data and/or events
  - Crowdsensing

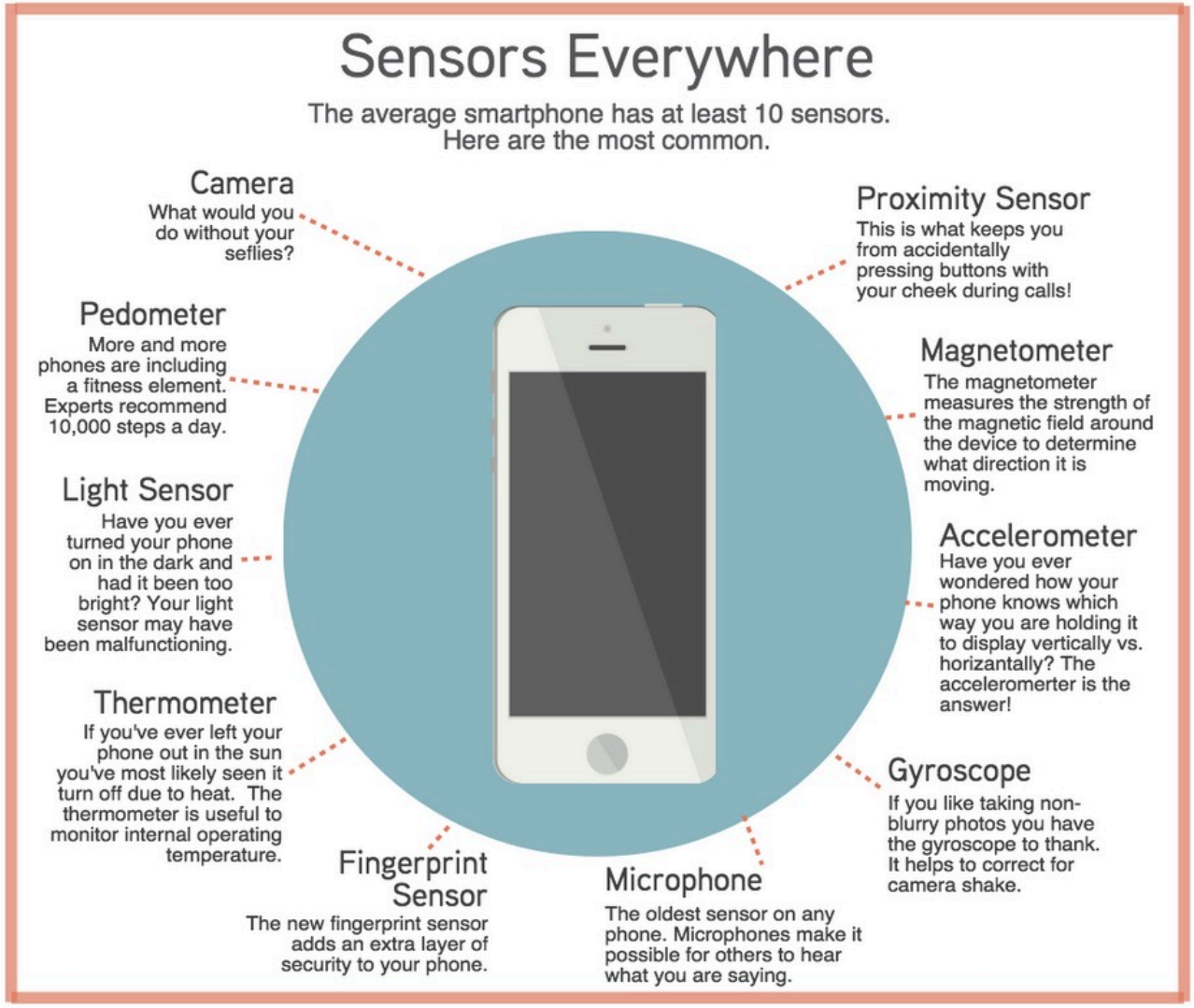
https://www.waze.com







Endpoints have sensors… tons of sensors



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

Power sources

- 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

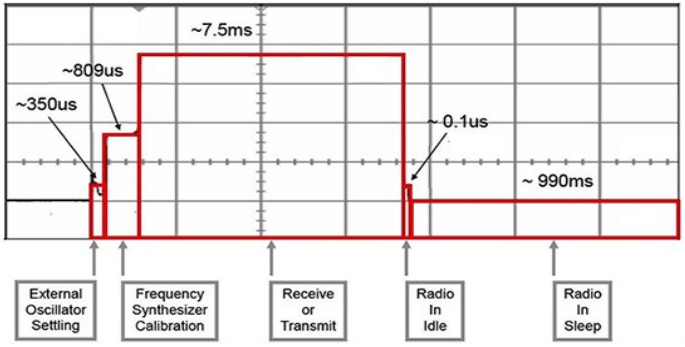
Lithium Ion (Li-ion)
Lithium-Ion Polymer (Li-Po)

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 µW
- 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

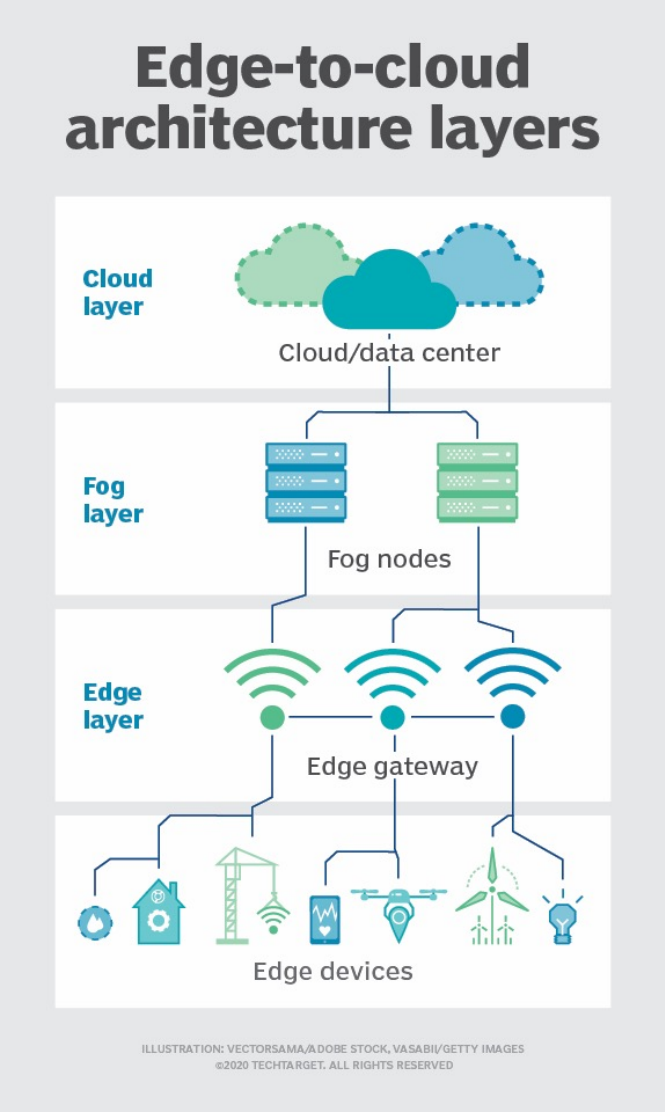
- **Of the hardware components of a "thing", the radio is usually the most power-consuming component.**
  - Compared to the power consumption of the microcontroller or the sensors, the radio transceiver often uses ten times as much power.
- This is due to the processing required for modulating and demodulating the radio signal.
  - 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.



What does that all mean?

So… Edge Computing

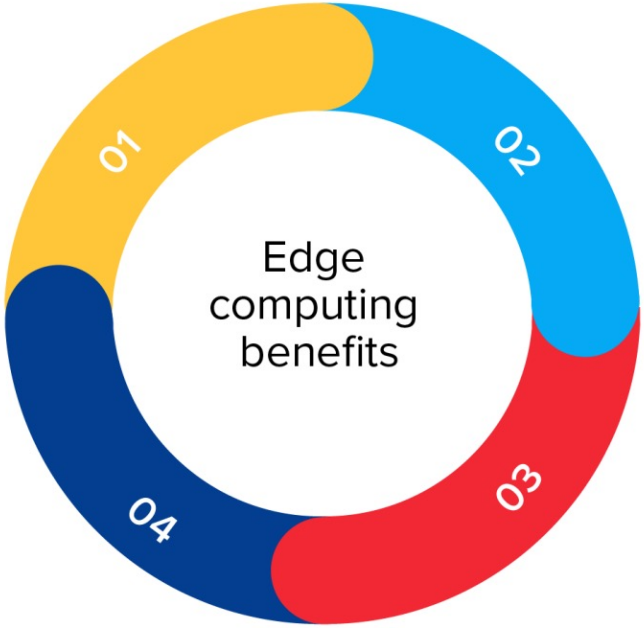
- A possible definition of edge computing is: "a part of a distributed computing topology in which information processing is located close to the edge — where things and people produce or consume that information."



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

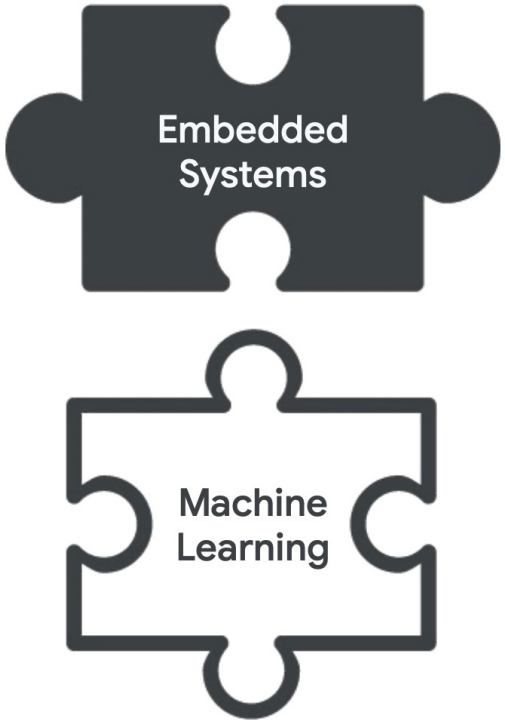


Bandwidth
Sending data from edge to the cloud takes up spectral resources; there's just not enough bandwidth for data transportation

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?



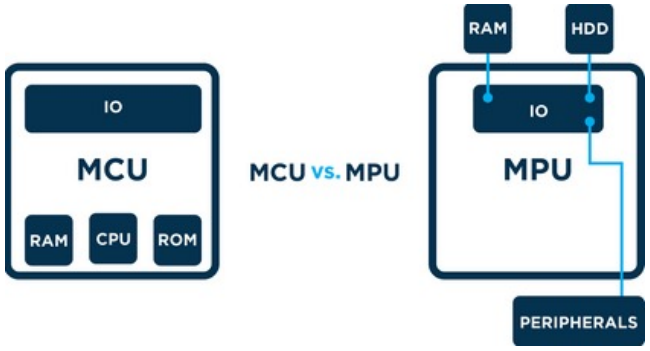
Original definition: "…a neural network model that runs at an energy cost of below 1 mW."
© "TinyML" by Pete Warden, Daniel Situnayake

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."

Microcontroller

- The microcontroller gives "things" their intelligence. It runs the software of the devices and is also responsible for connecting the radio with the sensors and actuators.
- 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.



Orders of magnitude

	Microprocessor	>	Microcontroller
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

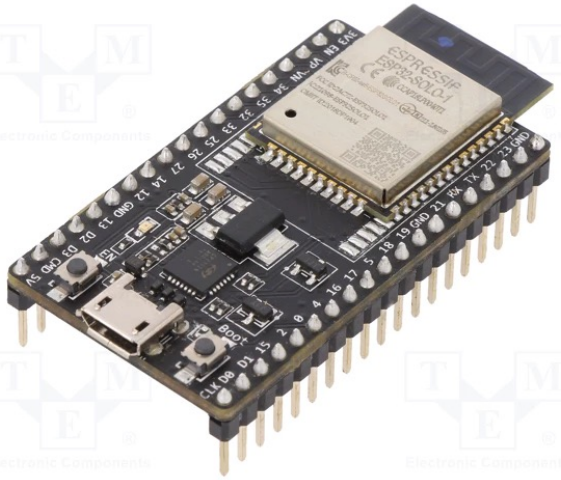
Microcontroller: ESP32

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

http://esp32.net

- 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

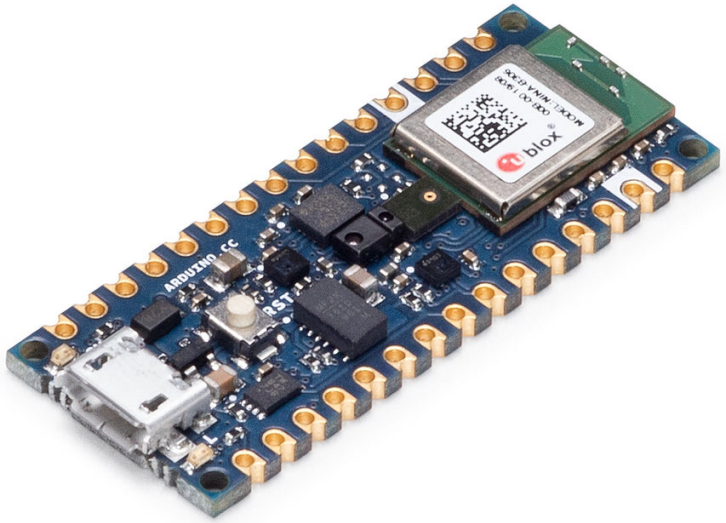
- CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS
- Memory: 520 KiB SRAM
- Wireless connectivity:
  - Wi-Fi: 802.11 b/g/n
  - Bluetooth: v4.2 BR/EDR and BLE
- Peripheral interfaces:
  - 12-bit SAR ADC up to 18 channels
  - 2 × 8-bit DACs
  - 10 × touch sensors (capacitive sensing GPIOs)
  - …



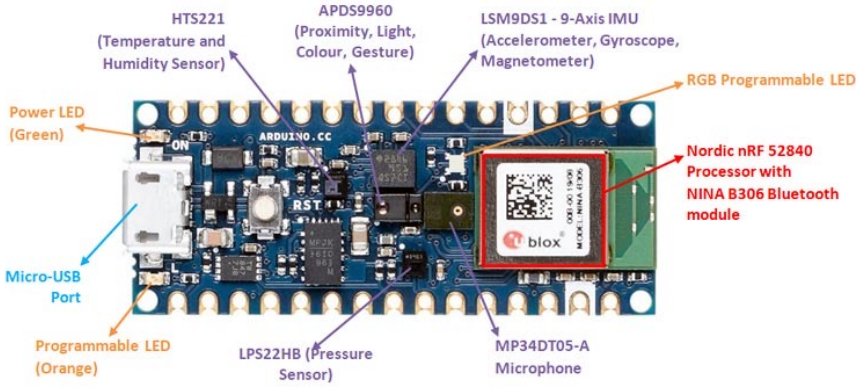
Mode	Min	Typ	Max	Unit
Transmit 802.11b, DSSS 1 Mbps, POUT = +19.5 dBm	-	240	-	mA
Transmit 802.11g, OFDM 54 Mbps, POUT = +16 dBm	-	190	-	mA
Transmit 802.11n, OFDM MCS7, POUT = +14 dBm	-	180	-	mA
Receive 802.11b/g/n	-	95 ~ 100	-	mA
Transmit BT/BLE, POUT = 0 dBm	-	130	-	mA
Receive BT/BLE	-	95 ~ 100	-	mA

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

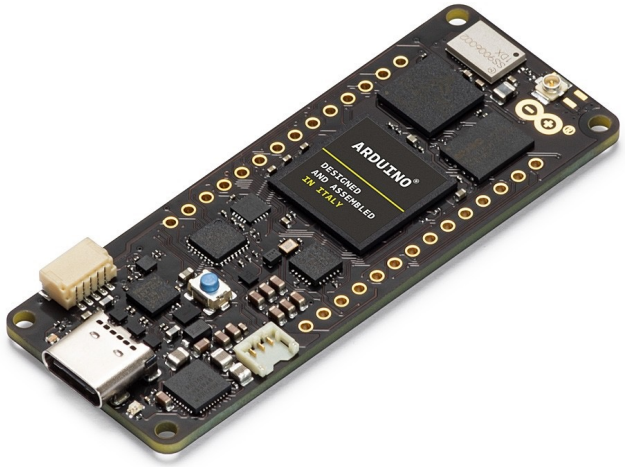
Most widely used HW for TinyML (currently)

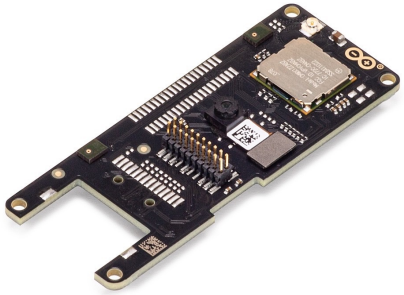


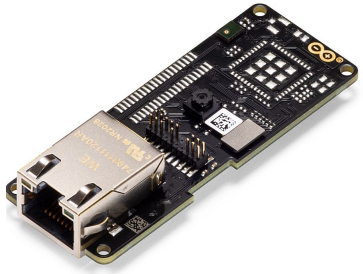
Arduino Nano 33 BLE Sense
64 MHz Arm® Cortex-M4F (with FPU)
1 MB Flash + 256 KB RAM
Bluetooth® 5 multiprotocol radio



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.

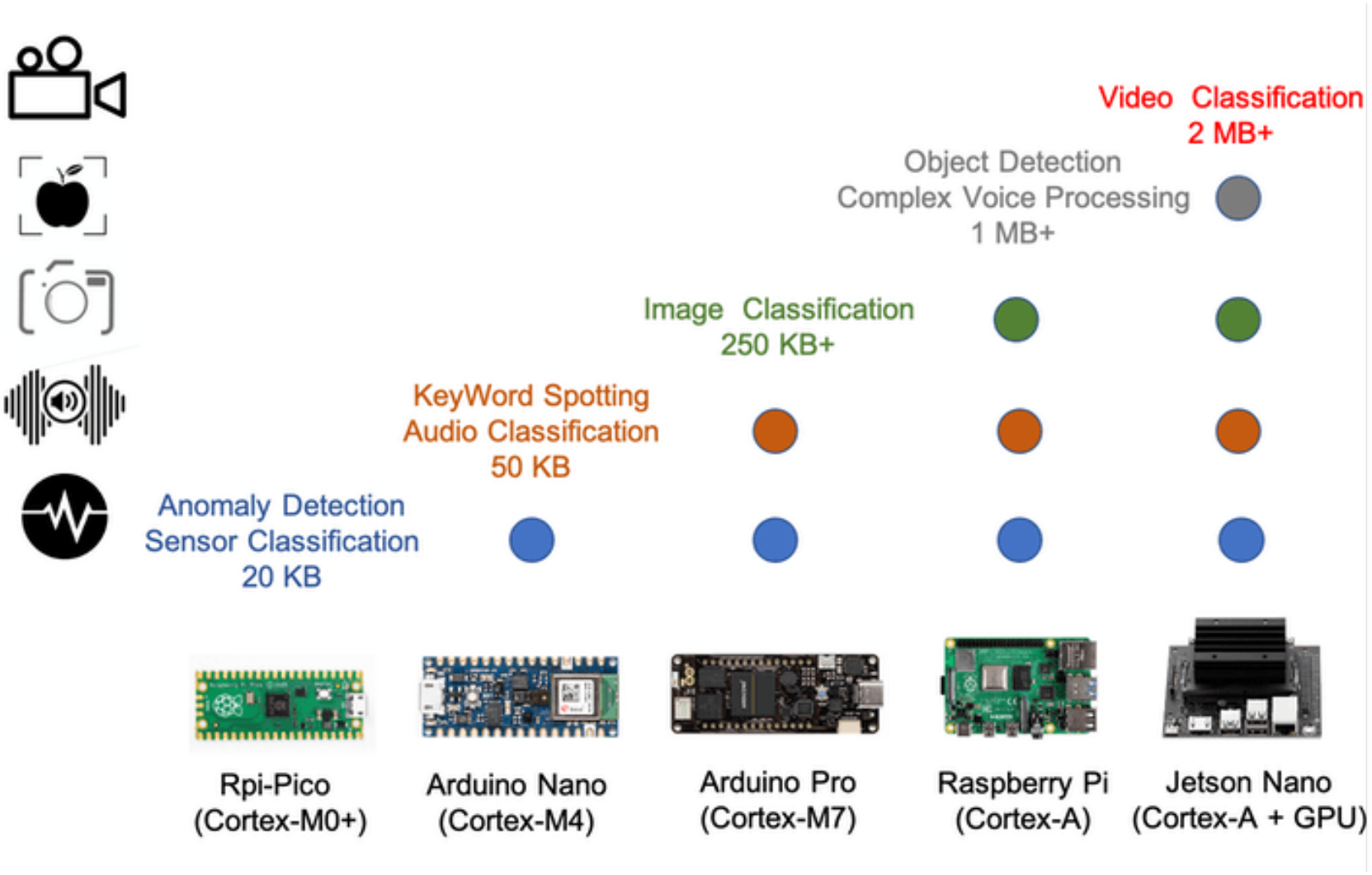
Both processors share all the in-chip peripherals and can run:

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

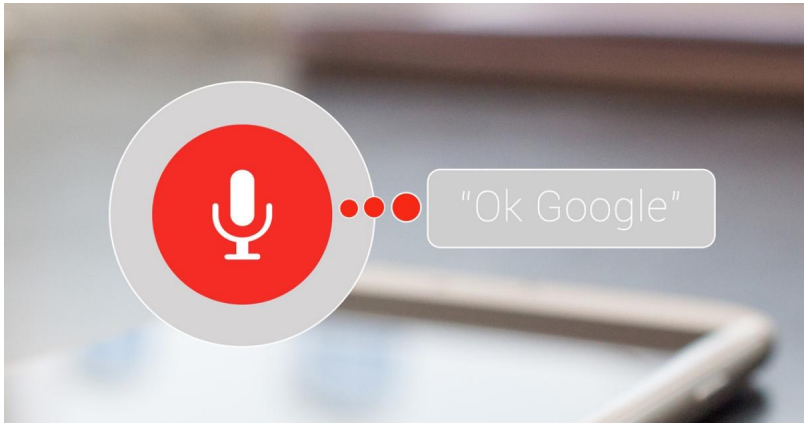
Or even… https://ouraring.com/

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

Current limitations

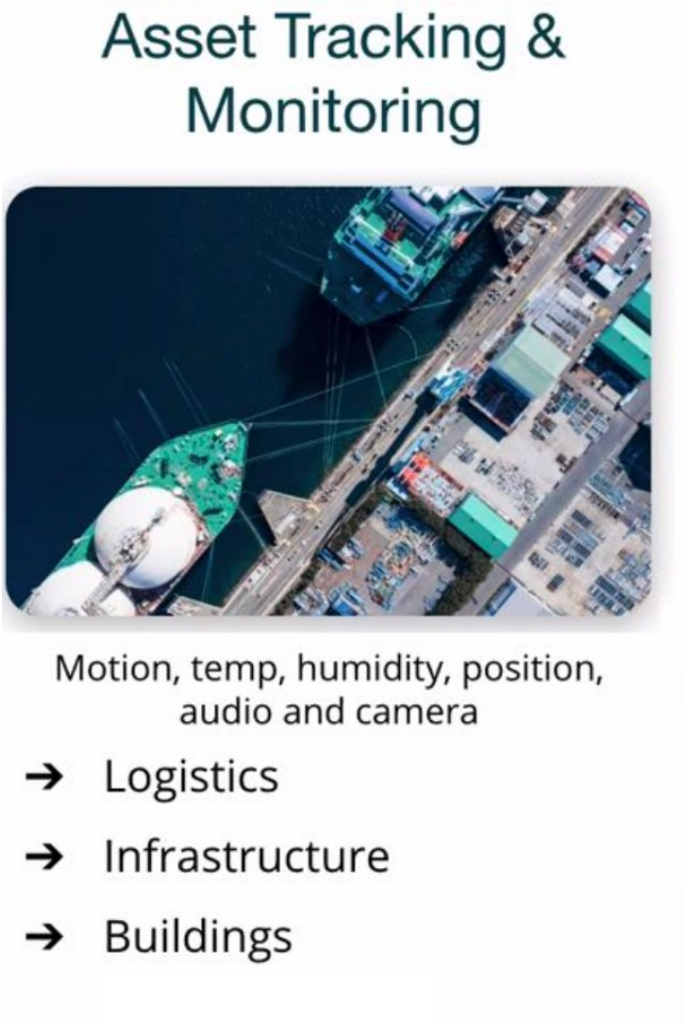


Applications of TinyML: the beginning



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



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…

providing solutions to farmers and extension workers by leveraging advances in AI, mobile phones, drones, satellites and nanotechnology

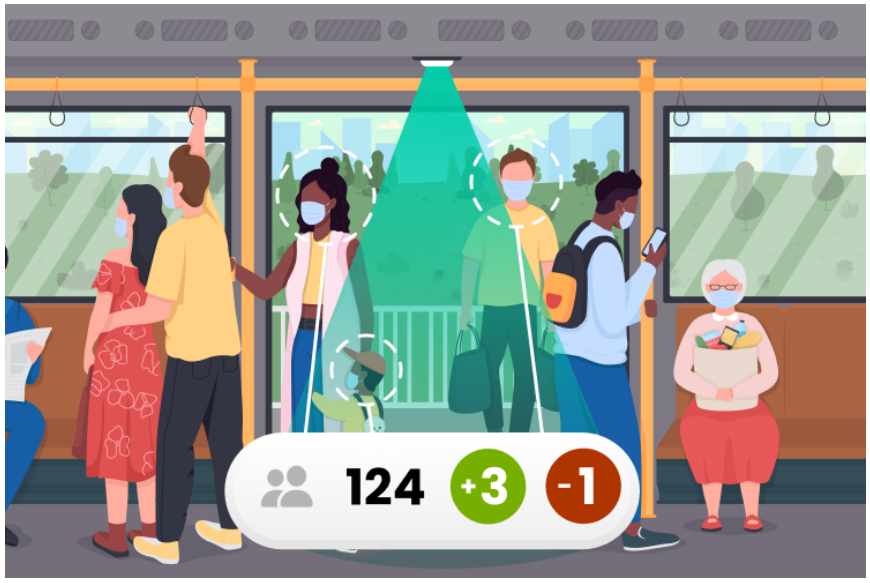
Plant Village: https://plantvillage.psu.edu

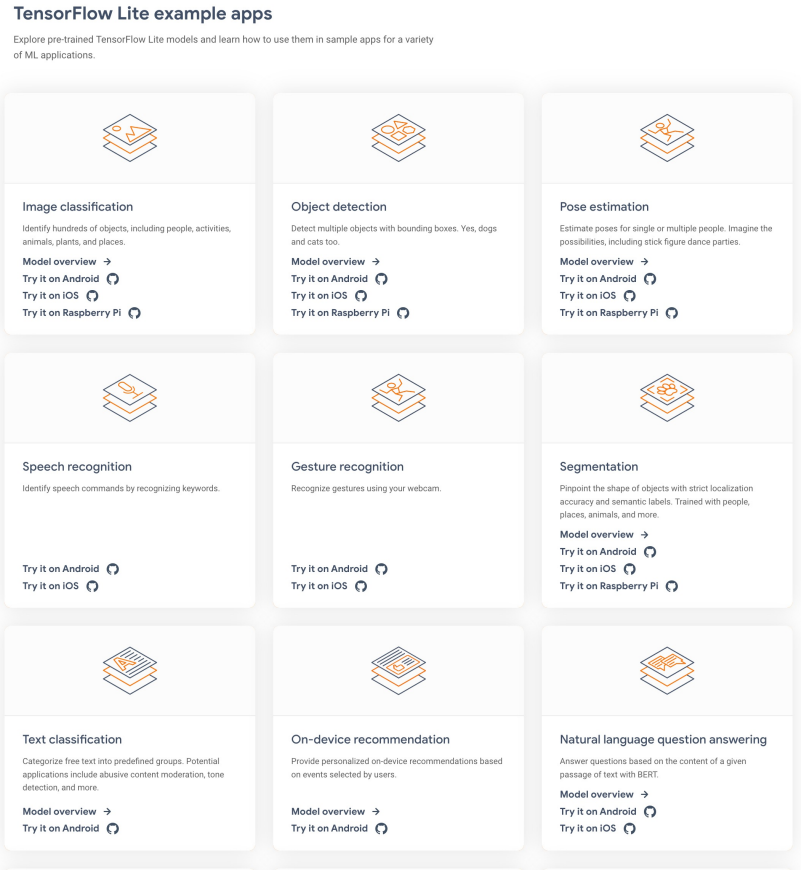
But the range of applications is wide…

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

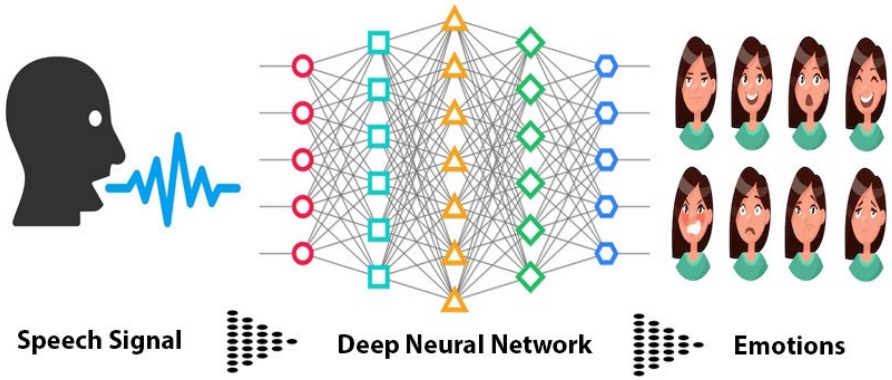
And...

People counting

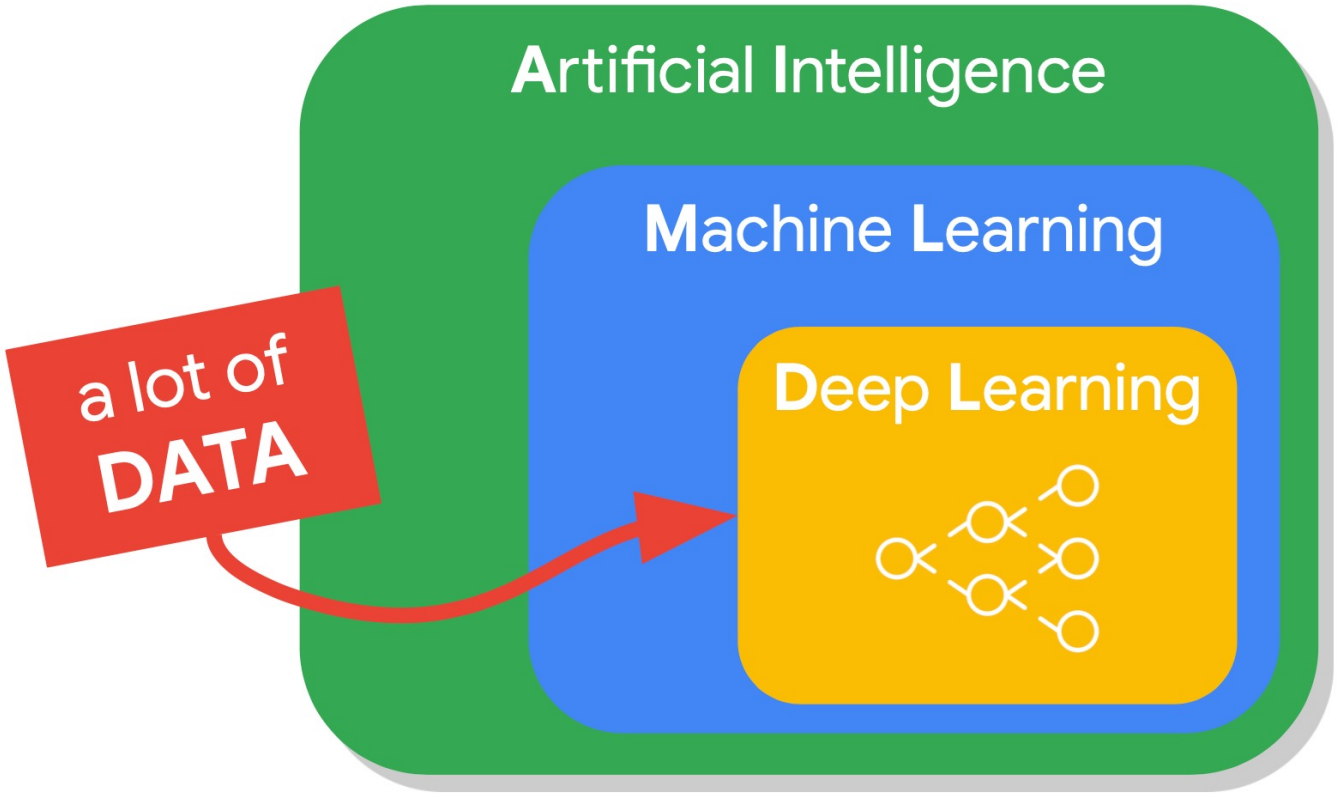




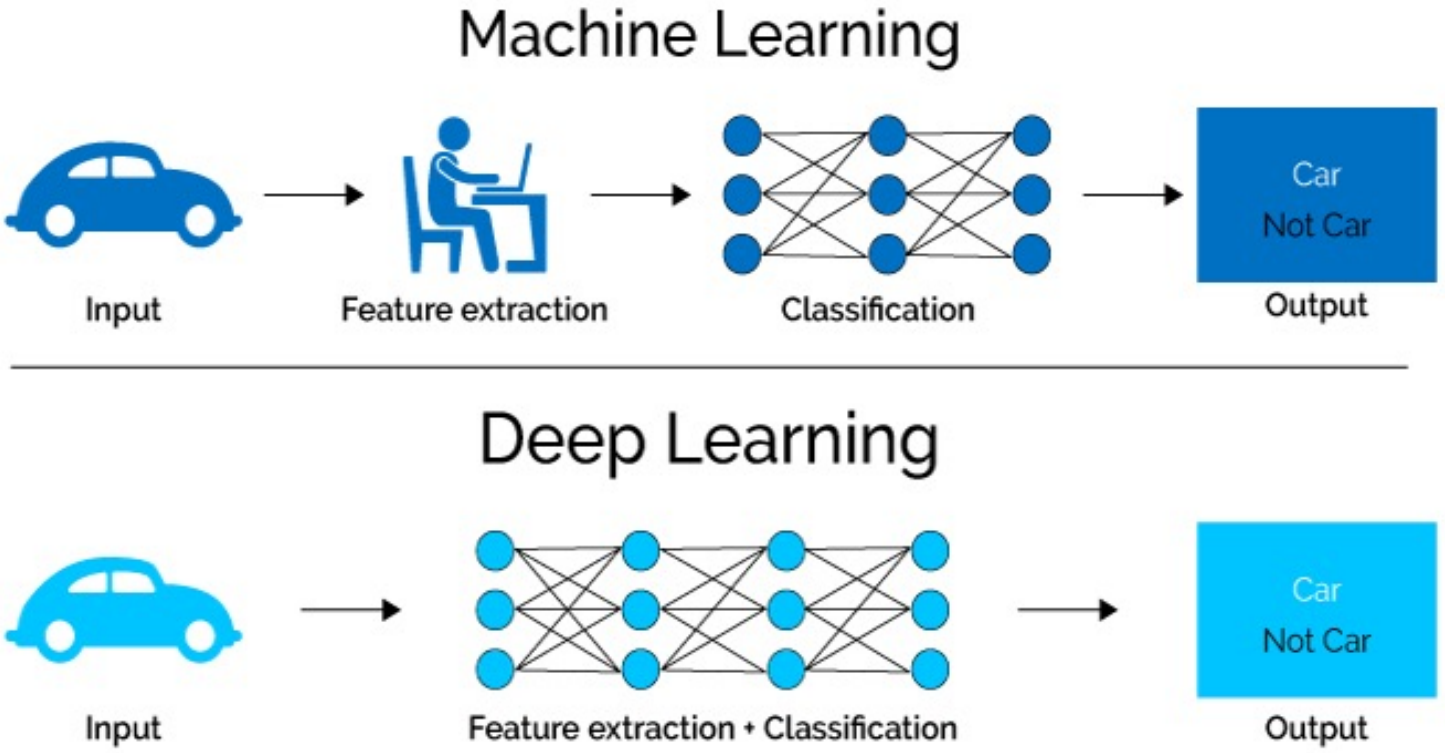
Sentiment analysis



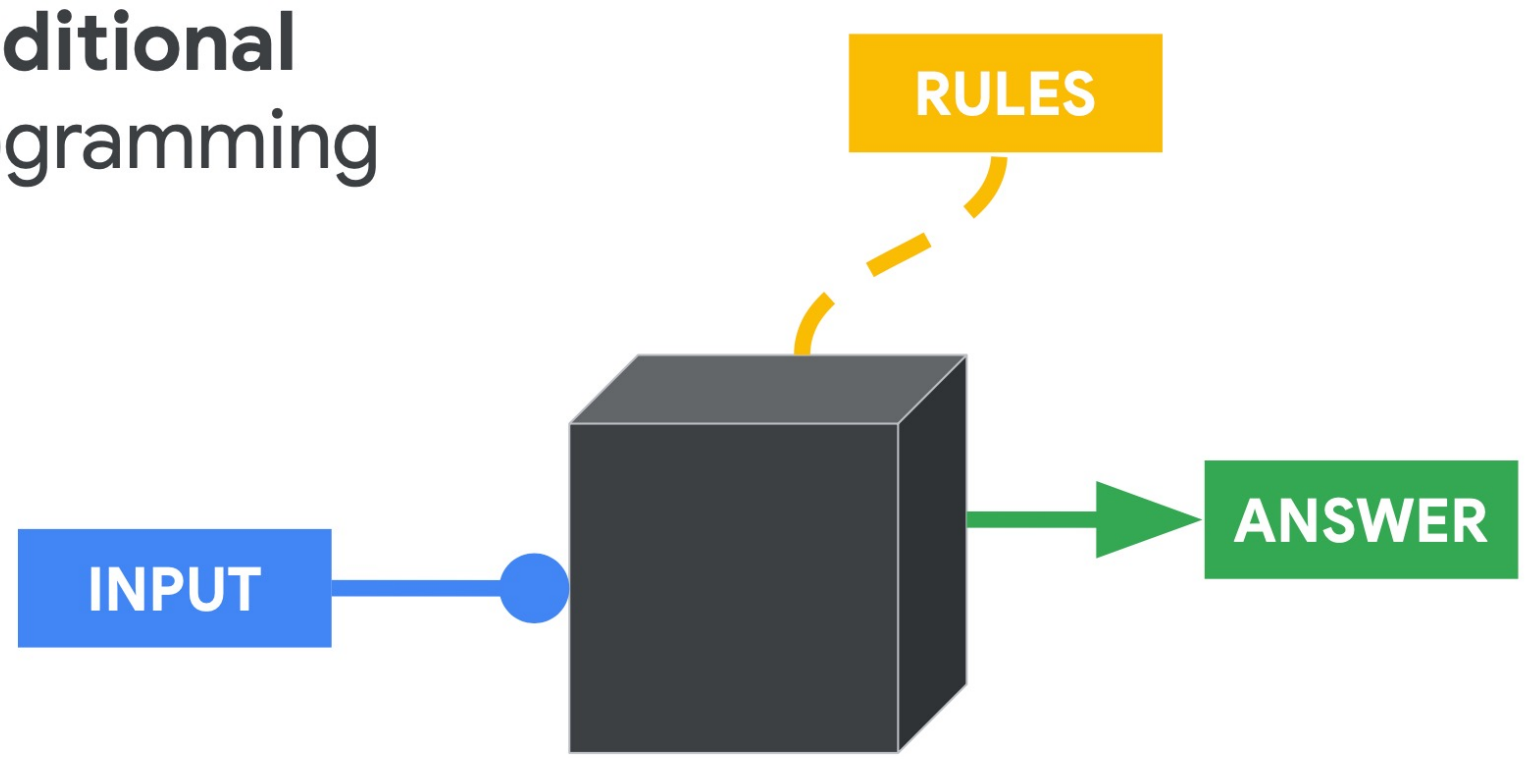
Machine Learning: Some initial definitions



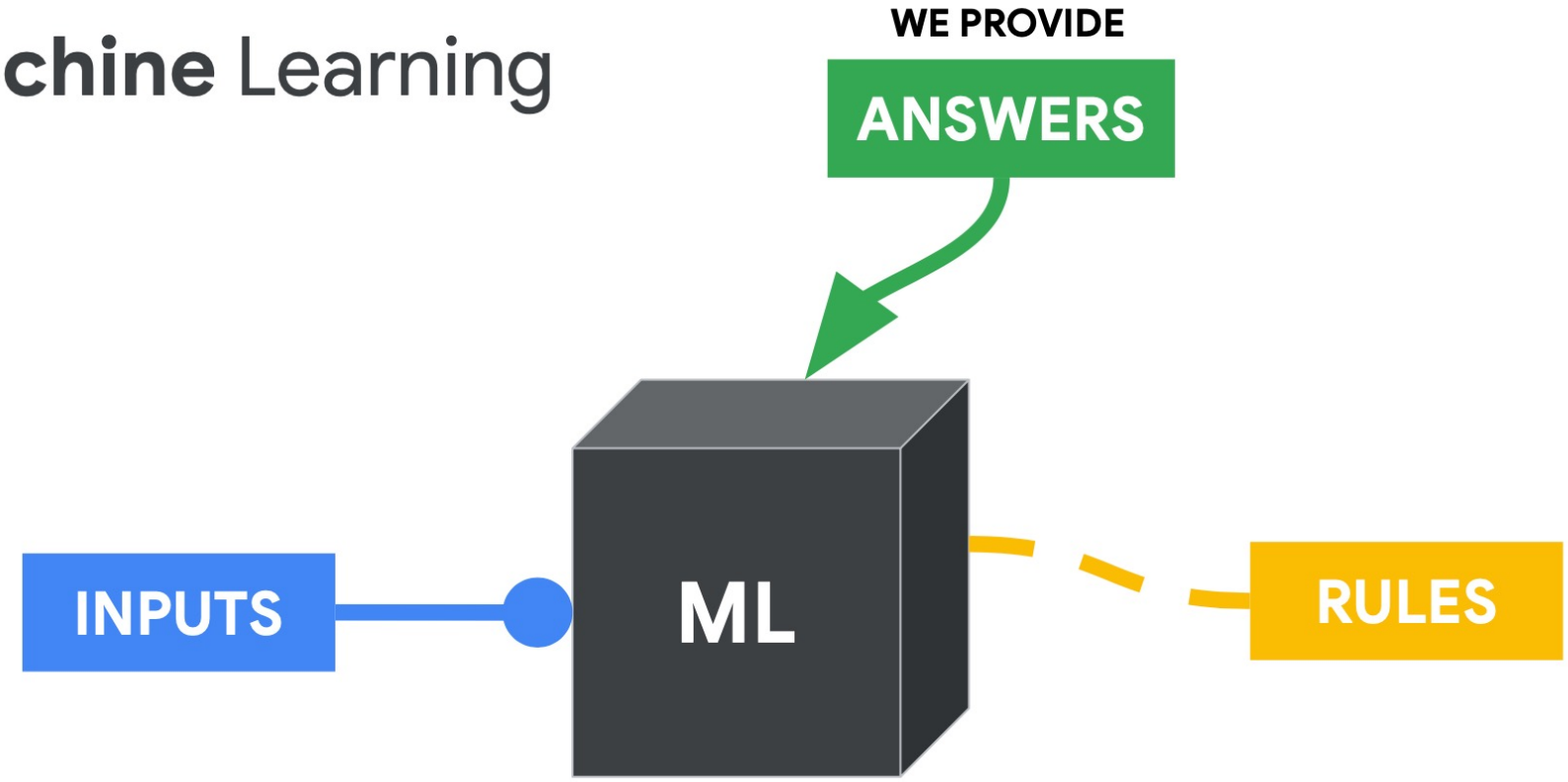
Machine Learning vs Deep Learning



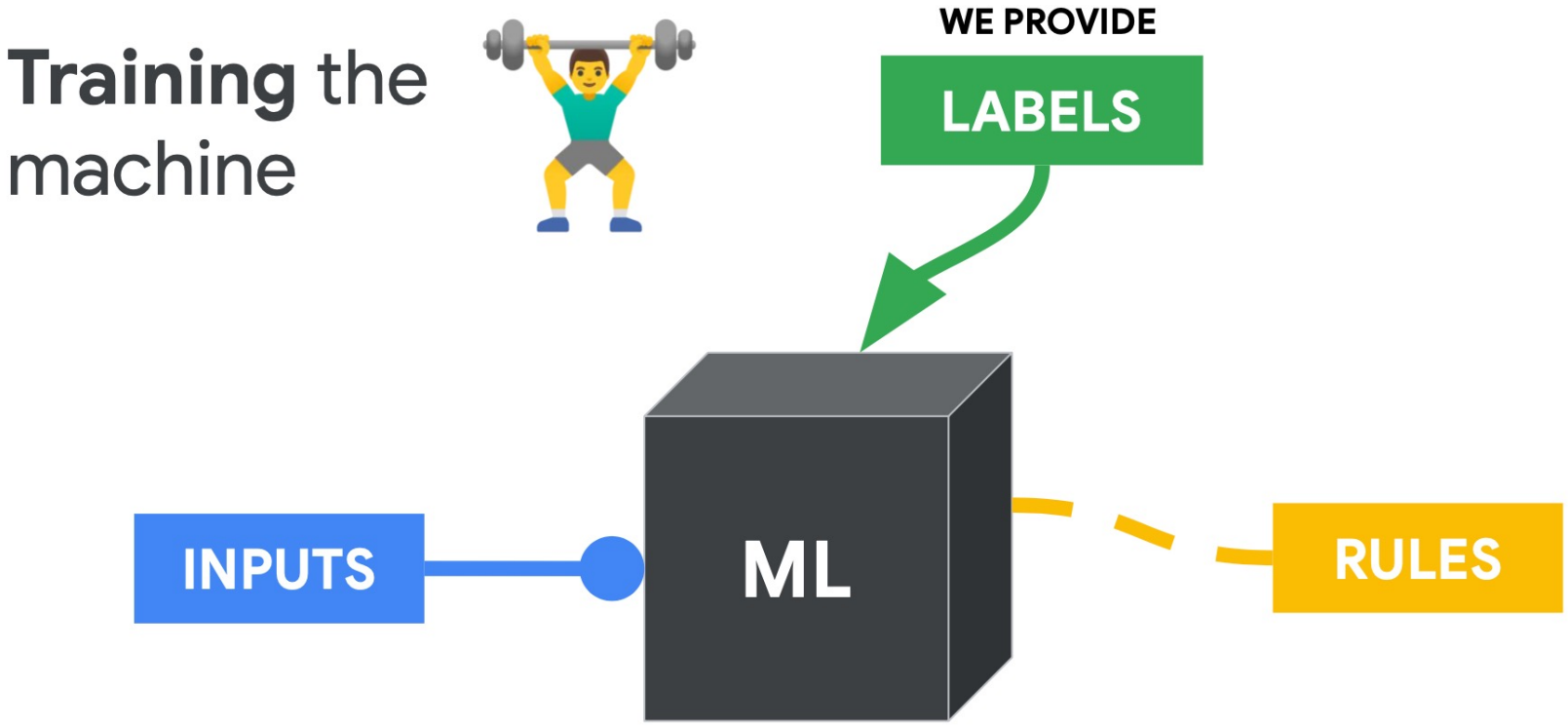
Traditional programming



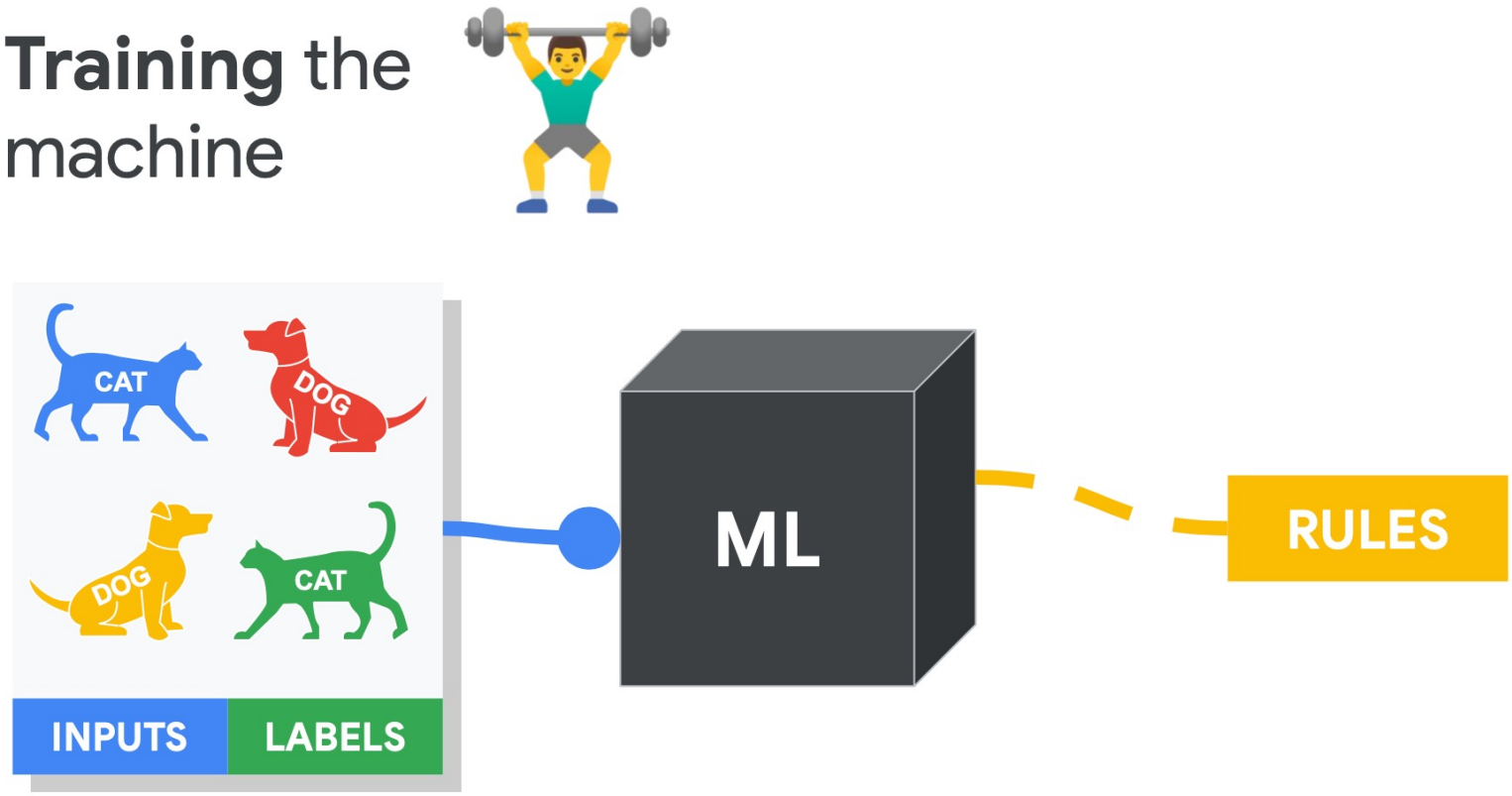
Machine learning



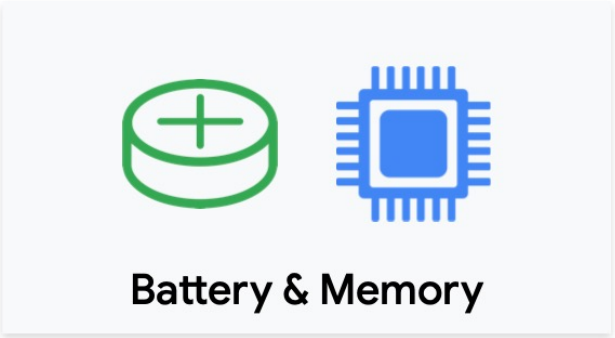
Training

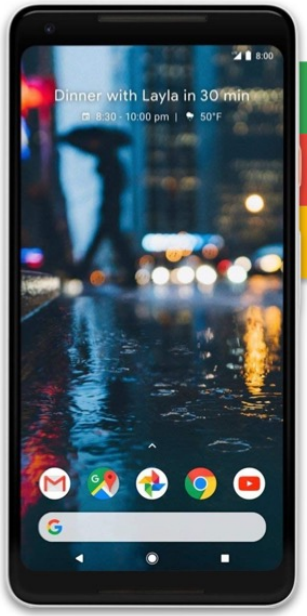


Training



Deploying





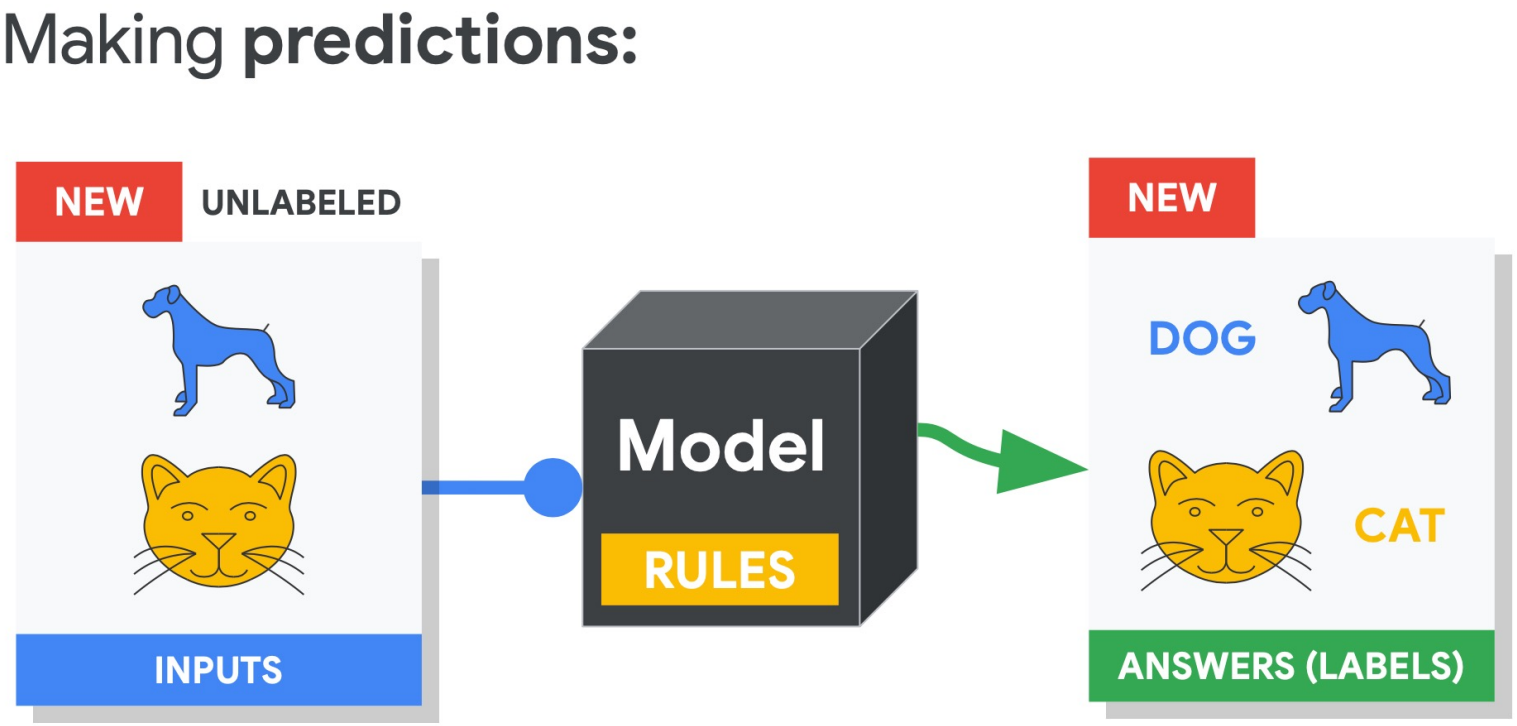
Less memory

Less compute power

Only focused on *inference*

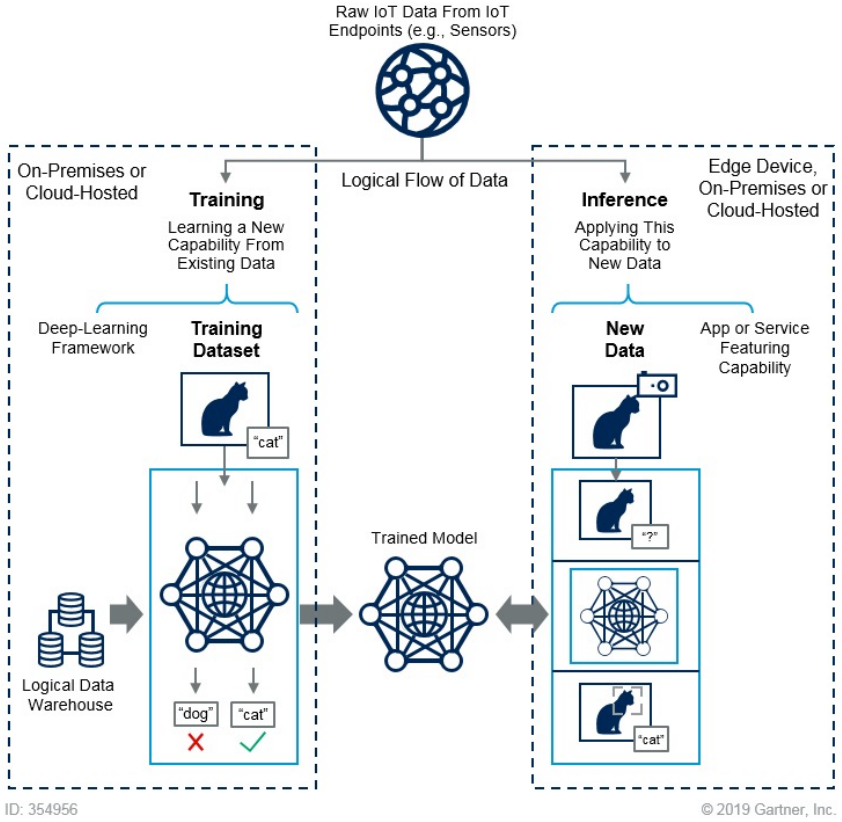
Collect & Transform Data → Design & Train a Model → **Deploy Model** → Make Inferences

Predictions / Inferencing



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.



Machine Learning algorithms

- Machine Learning algorithms are generally categorized based upon the type of output variable and the type of problem that needs to be addressed.
- These algorithms are broadly divided into three types i.e., **Regression, Clustering, and Classification.**
  - Regression and Classification are types of supervised learning algorithms while Clustering is a type of unsupervised algorithm.
- Clustering algorithms are generally used when we need to create the clusters based on the characteristics of the data points.
- **When the output variable is continuous, then it is a regression problem whereas when it contains discrete values, it is a classification problem.**

Basic types of classification algorithms

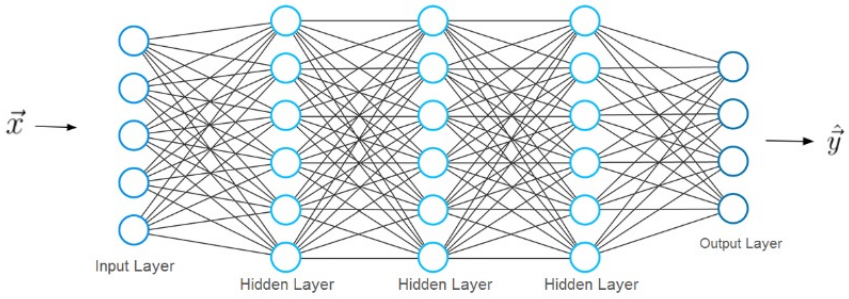
- **Logistic Regression**: a linear model; It uses the sigmoid function to calculate the probability of a certain event occurring. It is an ideal method for the classification of binary variables.

- **K-Nearest Neighbours (kNN)**: – It uses distance metrics like Euclidean distance, Manhattan distance, etc. to calculate the distance of one data point from every other data point. To classify the output, it takes a majority vote from k nearest neighbors of each data point.

- **Random Forest**: It is an ensemble learning method that involves multiple decision trees to predict the outcome of the target variable. Each decision tree provides its own outcome. In the case of the classification problem, it takes the majority vote of these multiple decision trees to classify the outcome. In the case of the regression problem, it takes the average of the values predicted by the decision trees.

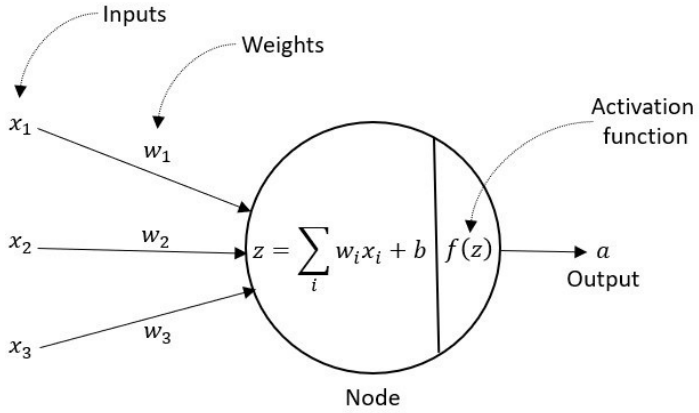
Artificial Neural Networks

- ANN are made of neurons (also called "nodes") organized in layers with an input and an output dimension.
- The individual layers of neural networks can also be thought of as a sort of filter that works from gross to subtle, increasing the likelihood of detecting and outputting a correct result.
- The layers can be grouped as:
  - **Input layer** has the job to pass the input vector to the Neural Network. If we have an input with 3 features, this layer takes 3 numbers as the input and passes the same 3 numbers to the next layer
  - **Hidden layers** represent the intermediary nodes, they apply several transformations to the numbers in order to improve the accuracy of the result
  - **Output layer** that returns the final output of the Neural Network.
    If we are doing a simple binary classification or regression, the output layer shall have only 1 neuron (so that it returns only 1 number).



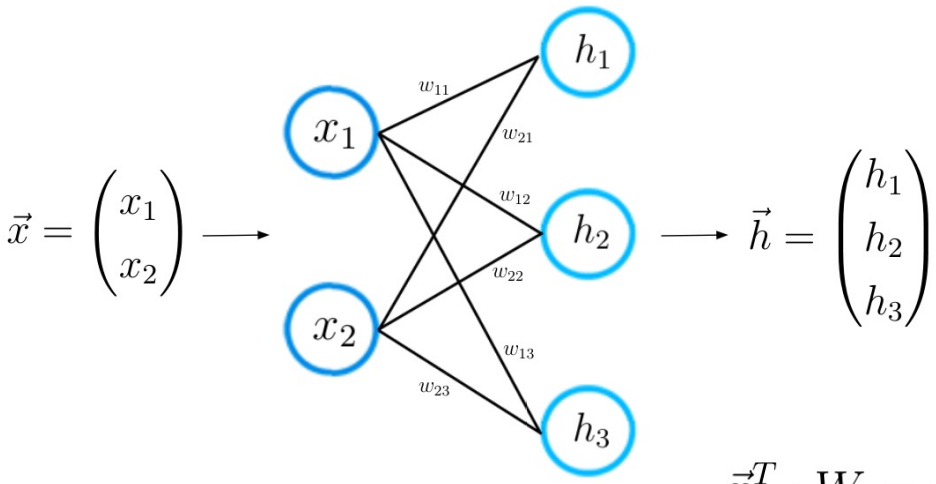
Artificial Neural Networks

- A node is a computational unit that connects the weighted inputs through an activation function
- Each neuron contains a '**bias**' value, and each connection contains a '**weight**' value.
- In the training phase, the model tries to find the best values for weights and biases to minimize the loss function and achieve better results.
- The weights are randomly initialized and optimized during the training to minimize a loss function.
- NN are not normally used directly with the raw data (such as .csv, images, text, etc.). We need a preprocessing step called **Feature Extraction.**



Forward propagation

$$W = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}$$



$$\vec{x}^T \cdot W = \begin{pmatrix} x_1, & x_2 \end{pmatrix} \cdot \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}$$

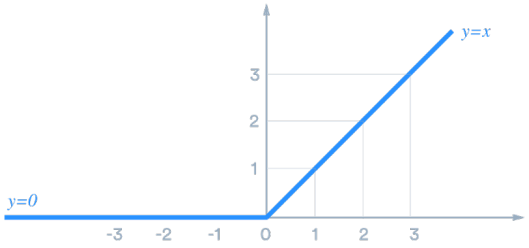
$$= \begin{pmatrix} x_1 w_{11} + x_2 w_{21}, & x_1 w_{12} + x_2 w_{22}, & x_1 w_{13} + x_2 w_{23} \end{pmatrix}$$

$$= \begin{pmatrix} z_1, & z_2, & z_3 \end{pmatrix} = \vec{z}\,, \quad \vec{h} = \sigma(\vec{z})$$

The final prediction vector ***h*** is obtained by applying a so-called activation function to the vector ***z***.
In this case, the activation function is represented by the letter **σ**

Activation function

- A neural network without an activation function is essentially just a linear regression model.
- Classic nonlinear activation functions are the **sigmoid**, the **hyperbolic tangent**, and the **Rectified linear unit (ReLU)**



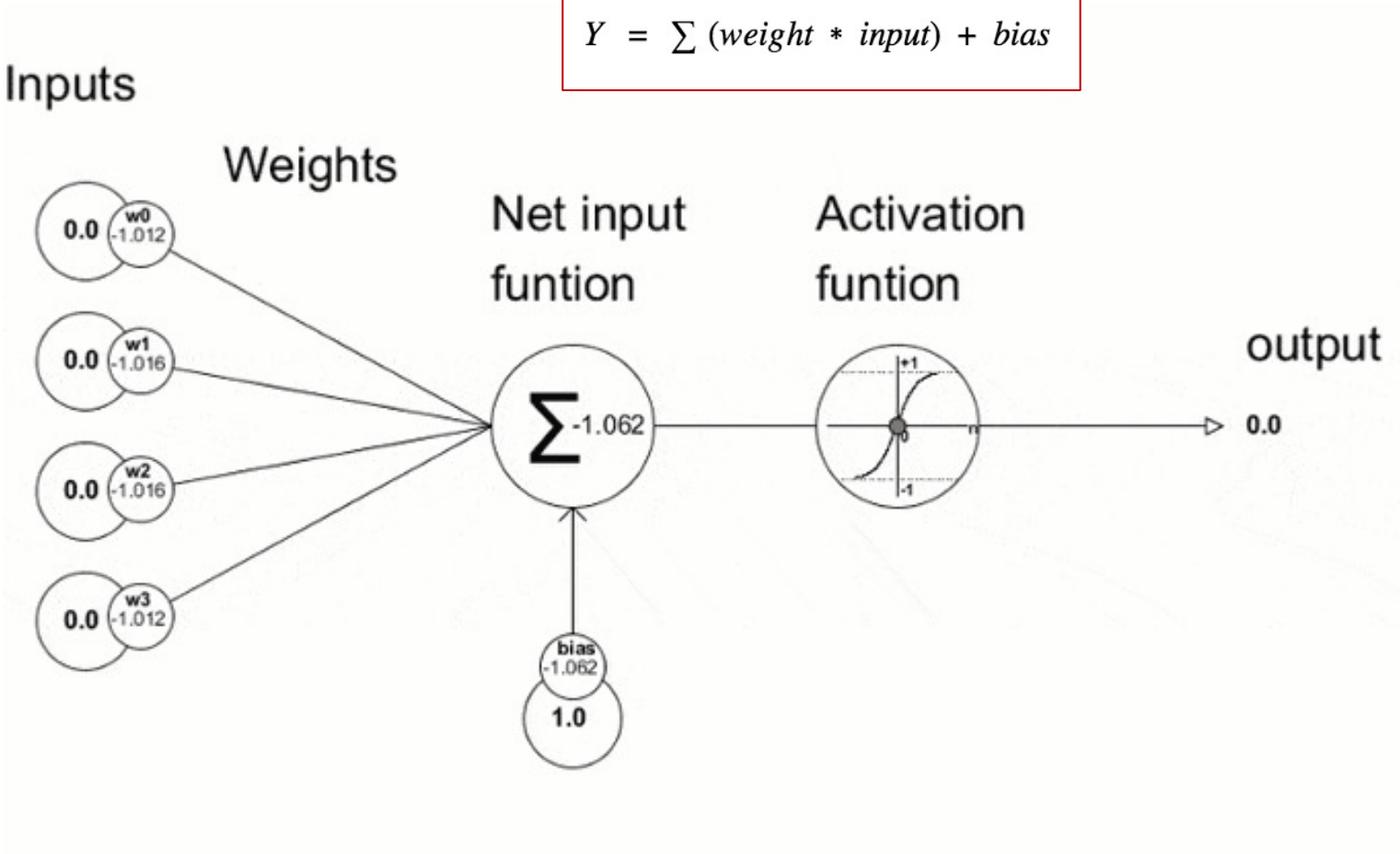

```
if input > 0:
  return input
else:
  return 0
```

- See for example: https://www.tensorflow.org/api_docs/python/tf/keras/activations

Activation function

- Activation functions are like filters, which controls how a node send data to the next ones (or if a node will be “fired” to send information, like the neurons in our brain).

$$Y = \sum (weight * input) + bias$$



Loss functions

Mean Squared Error (MSE) Loss:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2$$

y_i : entries in the prediction vector $\vec{y}$

$\hat{y}_i$: entries in the ground truth label $\hat{\vec{y}}$

Backward propagation: During gradient descent, we use the gradient of a loss function (or in other words the derivative of the loss function) to improve the weights of a neural network.

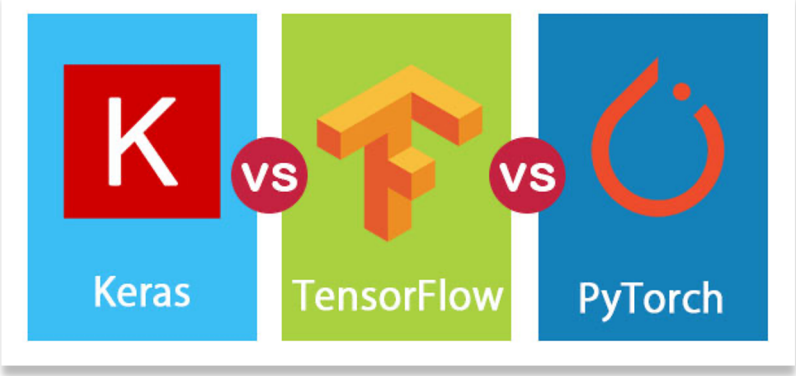
$$w_{11_{new}} = w_{11_{old}} - \epsilon \cdot \nabla_{w_{11}} \mathcal{L}(w_{11})$$

learning rate

- One **Epoch** is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE.
- Typically you can't pass the entire dataset into the neural net at once. So, you divide dataset into Number of **Batches**
- **Iterations** is the number of batches needed to complete one epoch.

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 PyTorch, TensorFlow, and Keras.
  - **Keras** is a neural network library scripted in Python and can execute on the top layer of TensorFlow.
  - **TensorFlow** is used to perform multiple tasks in data flow programming and machine learning applications.
  - **PyTorch** is a machine learning library that is mainly used in natural language processing.

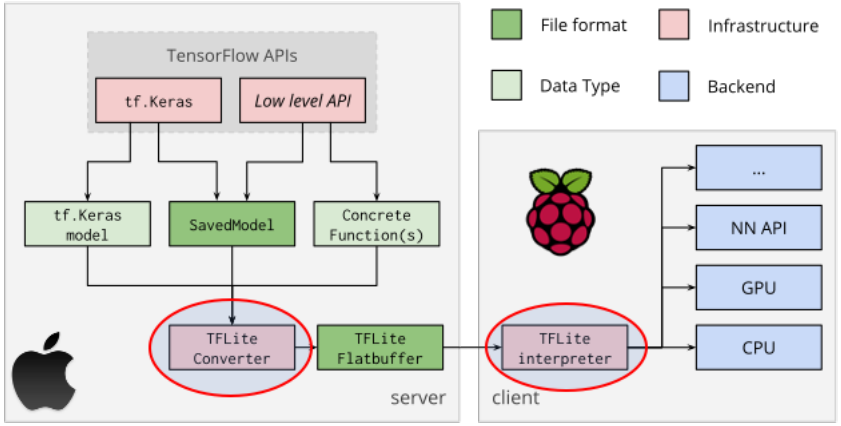


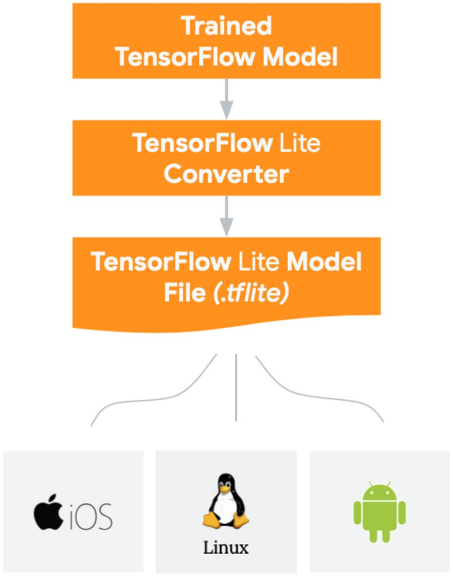
TensorFlow Lite para microcontroladores

- 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.
- While the field of TinyML is still relatively new and experimental, TensorFlow Lite Micro is one of the most popular frameworks being used, which means you'll be able to leverage a substantial amount of community experience and support in your own projects.

TensorFlow Lite (TFLite)

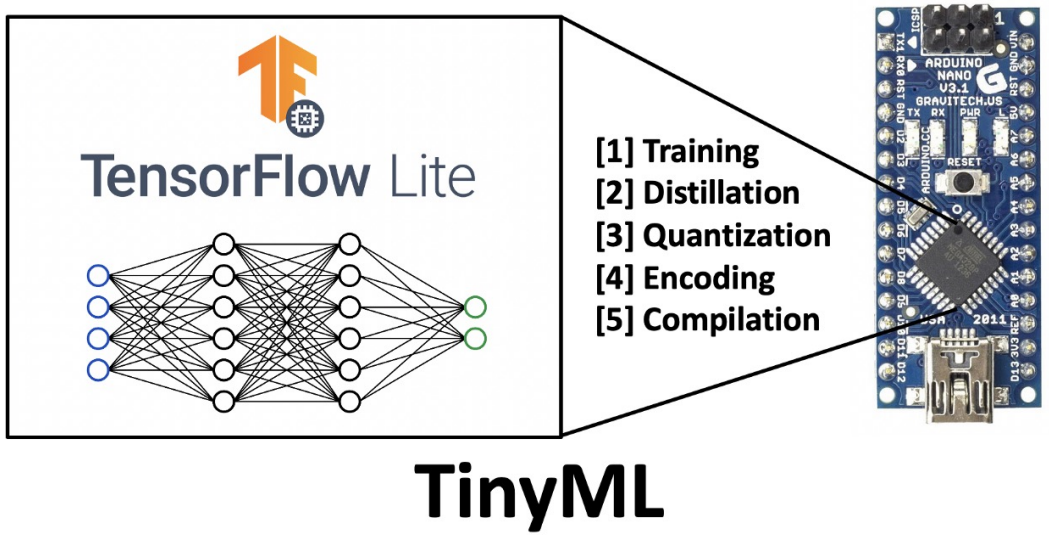
- **TensorFlow Lite (TFLite)** is an open-source deep learning framework that enables on-device machine learning inference.
- Consists of two main components:
  - The **TFLite converter**, which converts TensorFlow models into an efficient form for use by the interpreter and can introduce optimizations 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.





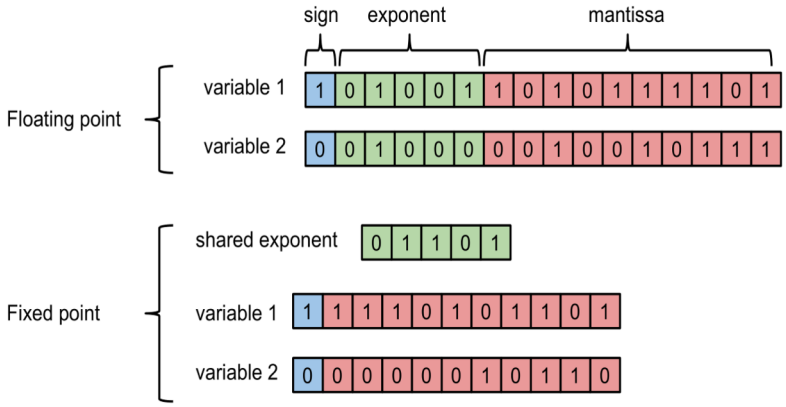
TensorFlow Lite optimizations

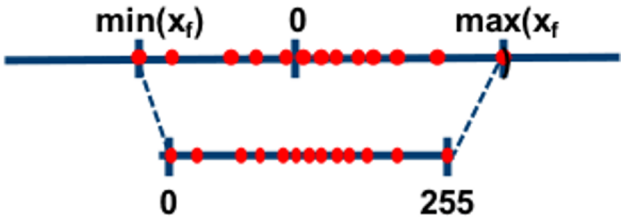
- Problem:
  - Less Compute Power, Less Memory, High Accuracy
- Some optimizations:
  - Quantization
  - Pruning
- ..and
  - Separable Convolutions
  - Knowledge Distillation



Quantization

- 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

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

Edge Impulse

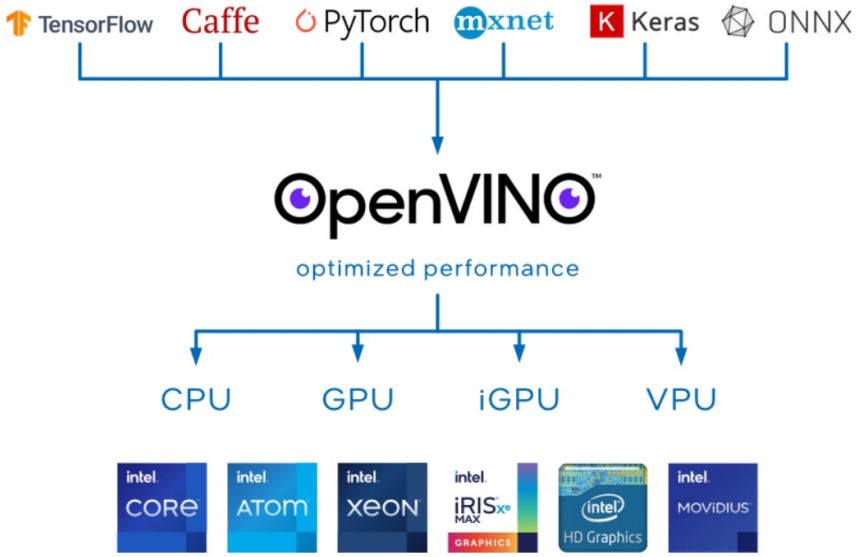
- http://edgeimpulse.com

- 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.

OpenVINO

https://docs.openvino.ai/

- 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



OpenMV

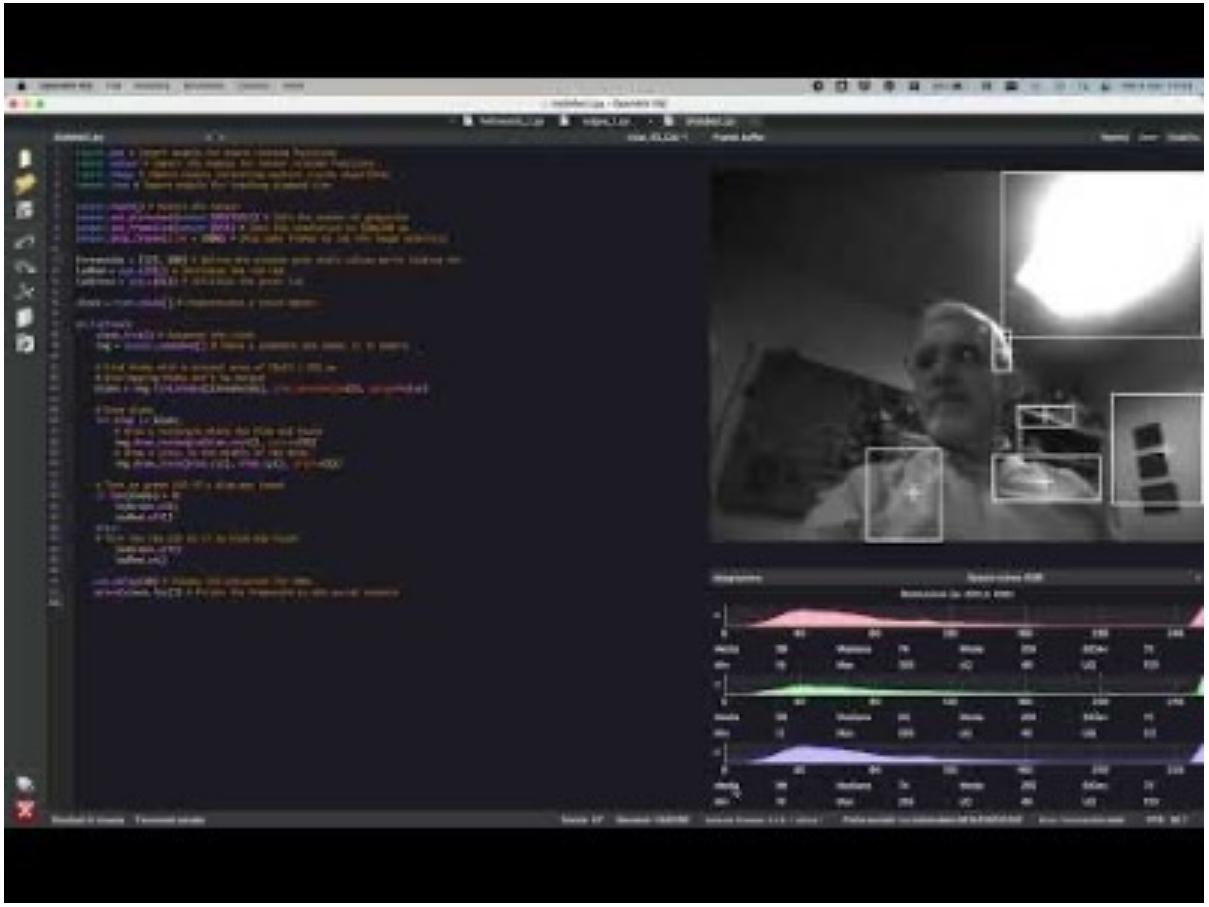
- http://openmv.io/

- 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!

Blob detection with Portenta

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



Blob detection with Portenta: the code

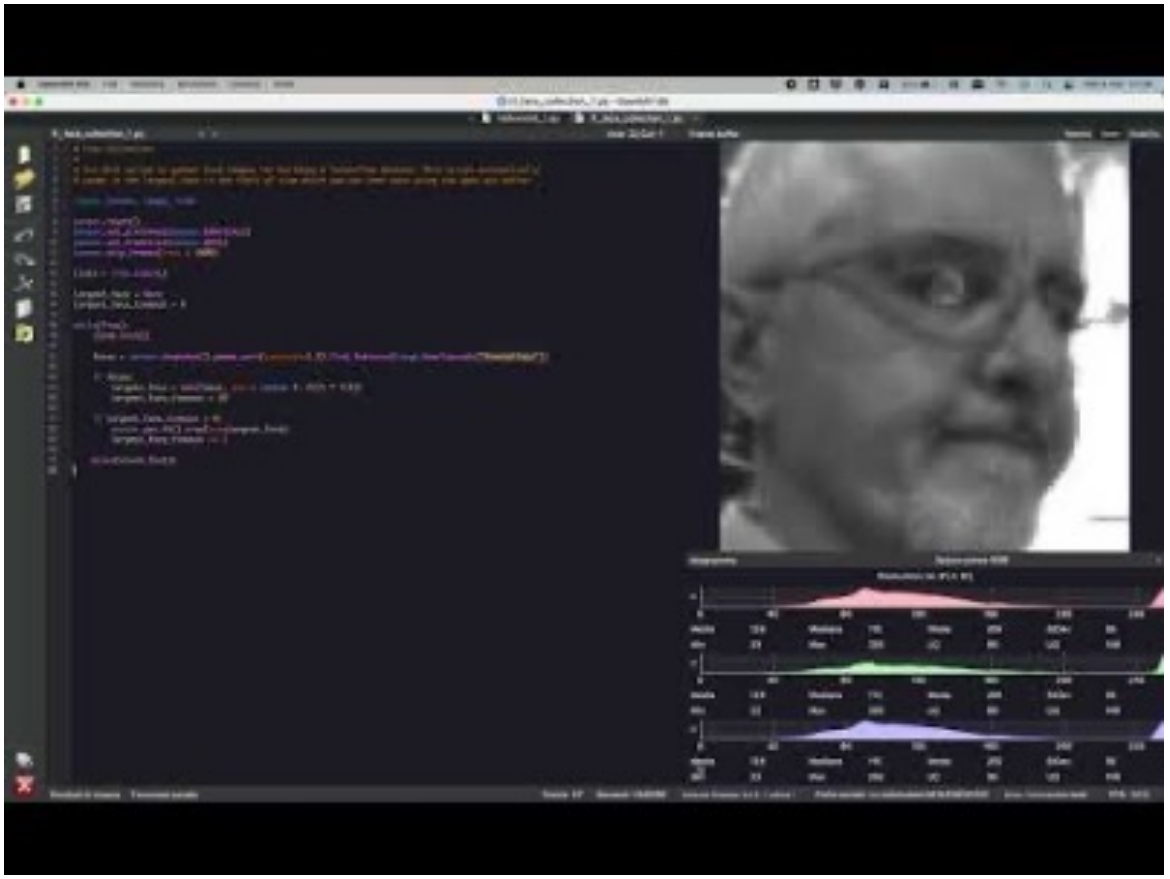
blobdect.py — UNREGISTERED

```python
import pyb # Import module for board related functions
import sensor # Import the module for sensor related functions
import image # Import module containing machine vision algorithms
import time # Import module for tracking elapsed time

sensor.reset() # Resets the sensor
sensor.set_pixformat(sensor.GRAYSCALE) # Sets the sensor to grayscale
sensor.set_framesize(sensor.QVGA) # Sets the resolution to 320x240 px
sensor.skip_frames(time = 2000) # Skip some frames to let the image stabilize

thresholds = (125, 200) # Define the min/max gray scale values we're looking for
ledRed = pyb.LED(1) # Initiates the red led
ledGreen = pyb.LED(2) # Initiates the green led

clock = time.clock() # Instantiates a clock object

while(True):
    clock.tick() # Advances the clock
    img = sensor.snapshot() # Takes a snapshot and saves it in memory

    # Find blobs with a minimal area of 15x15 = 225 px
    # Overlapping blobs won't be merged
    blobs = img.find_blobs([thresholds], area_threshold=225, merge=False)

    # Draw blobs
    for blob in blobs:
        # Draw a rectangle where the blob was found
        img.draw_rectangle(blob.rect(), color=255)
        # Draw a cross in the middle of the blob
        img.draw_cross(blob.cx(), blob.cy(), color=255)

    # Turn on green LED if a blob was found
    if len(blobs) > 0:
        ledGreen.on()
        ledRed.off()
    else:
    # Turn the red LED on if no blob was found
        ledGreen.off()
        ledRed.on()

    pyb.delay(50) # Pauses the execution for 50ms
    print(clock.fps()) # Prints the framerate to the serial console
```

Line 43, Column 1 — Spaces: 4 — Python

Face detection with Portenta

- This other example performs recognizes faces



tf_face_collection_1.py

Face detection with Portenta: the code

tf_face_collection_1.py — UNREGISTERED

tf_face_collection_1.py

```
# Face Collection
#
# Use this script to gather face images for building a TensorFlow dataset. This script automatically
# zooms in the largest face in the field of view which you can then save using the data set editor.

import sensor, image, time

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(time = 2000)

clock = time.clock()

largest_face = None
largest_face_timeout = 0

while(True):
    clock.tick()

    faces = sensor.snapshot().gamma_corr(contrast=1.5).find_features(image.HaarCascade("frontalface"))

    if faces:
        largest_face = max(faces, key = lambda f: f[2] * f[3])
        largest_face_timeout = 20

    if largest_face_timeout > 0:
        sensor.get_fb().crop(roi=largest_face)
        largest_face_timeout -= 1

    print(clock.fps())
```

Line 30, Column 1 — Spaces: 4 — Python

And finally: https://www.lobe.ai

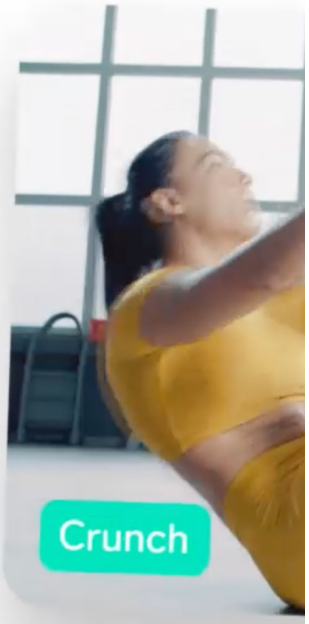
lobe

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



TinyML: Machine Learning meets the Internet of Things