We are learning on Noongar land

THE UNIVERSITY OF WESTERN AUSTRALIA

CITS 5506
The Internet of Things
Lecture 11
Tiny Machine Learning

Dr Atif Mansoor
atif.mansoor@uwa.edu.au

Lecture Overview

Ethical AI & Machine Learning

What Makes Tiny Machine Learning

Why TinyML is important

The Challenges of TinyML

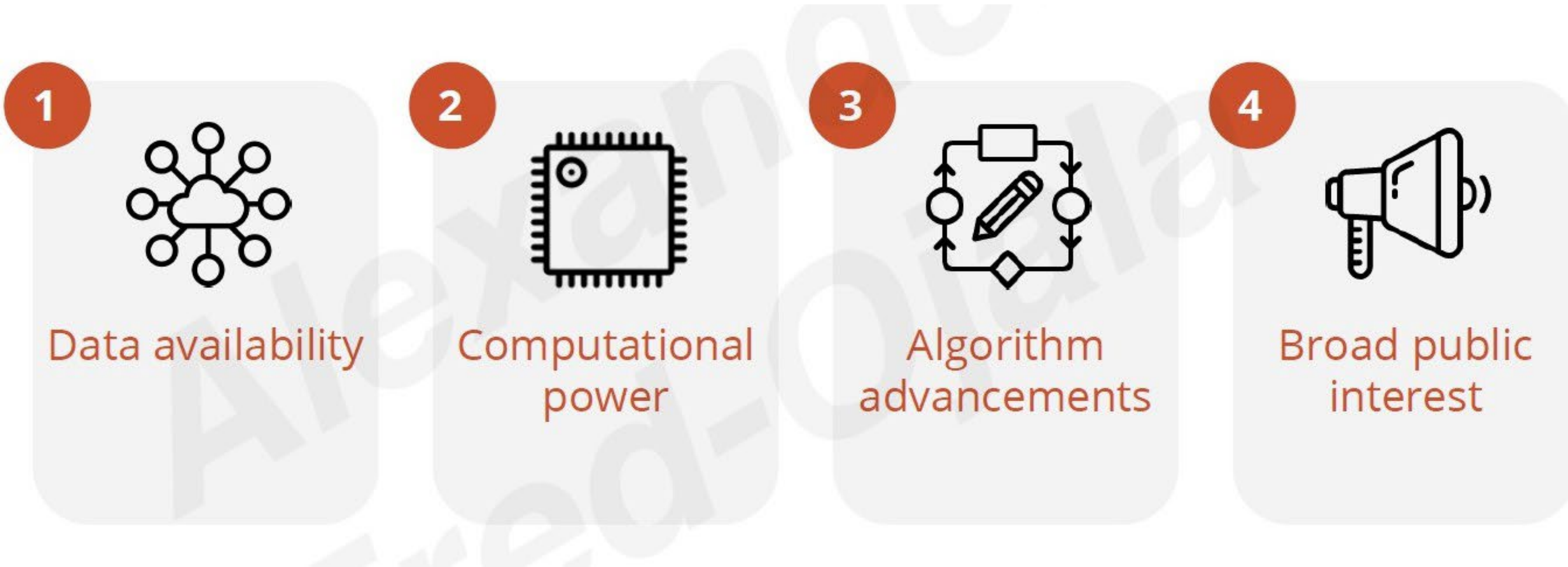
Model Evaluation : Performance Metrics

Ethical AI & Machine Learning

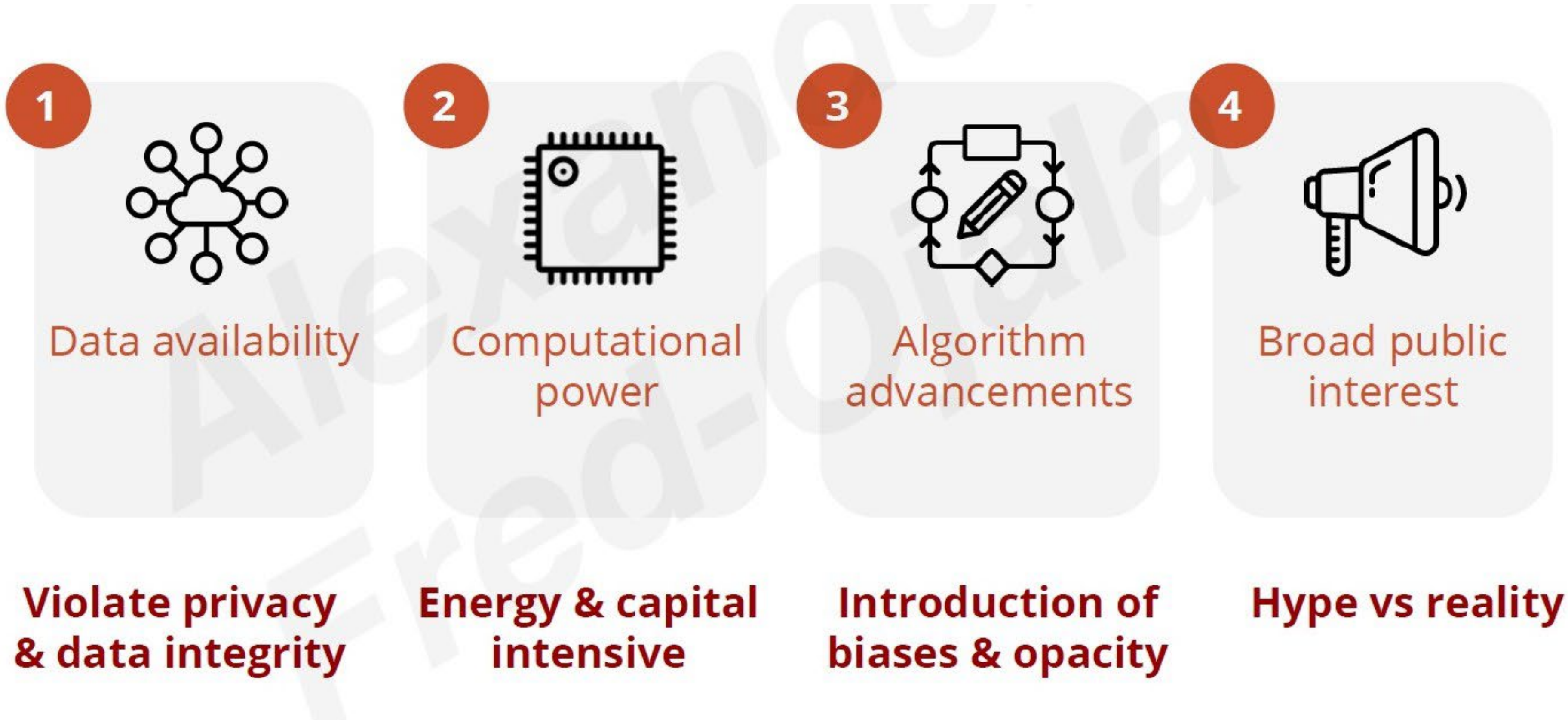
Desire:

Ethical, responsible and trustworthy AI

AI Enablers

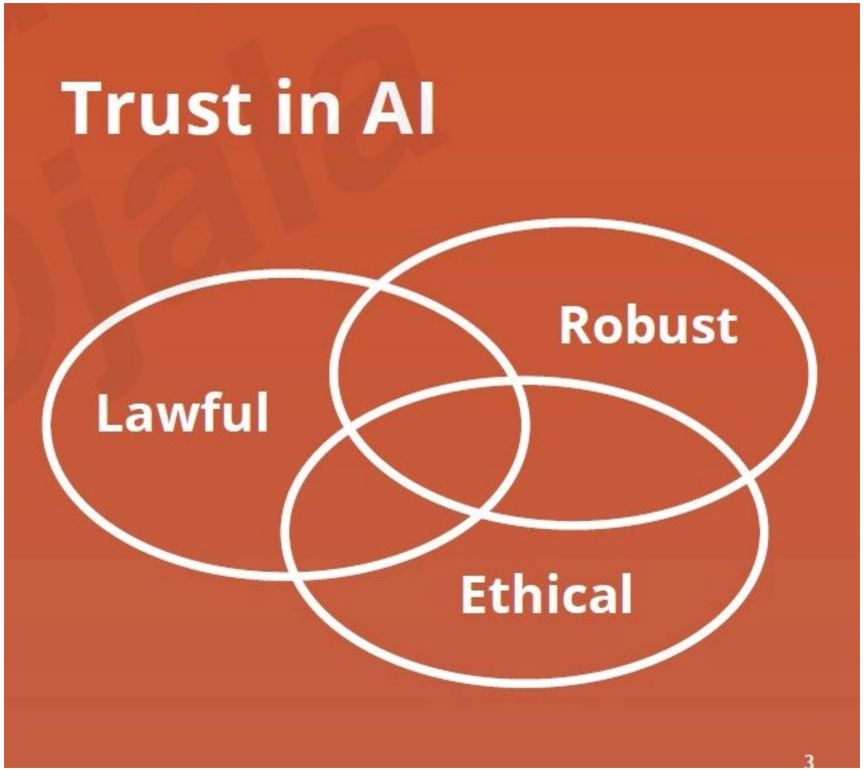

1
Data availability
2
Computational power
3
Algorithm advancements
4
Broad public interest

Flip side of AI Enablers



Trust in Humans vs Trust In AI

- Morals & Ethics
- Character
- Societal Laws
- Cultural Laws
- Compassion



Trustworthy AI

Trustworthy AI should be:

- Lawful - respecting all applicable laws and regulations
- Ethical - respecting ethical principles and values
- Robust - both from a technical perspective while taking into account its social environment (e.g fairness, inclusivity, alignment with social norms and values etc)

EU Ethics guidelines for trustworthy AI

- Human agency & oversight
- Technical robustness & safety
- Privacy & data governance
- Transparency
- Diversity, fairness & non-discrimination
- Societal & environmental wellbeing
- Accountability

https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

Human Agency and Oversight

- AI systems should empower human beings, allowing them to make informed decisions and fostering their fundamental rights.
- The allocation of functions between humans and AI systems should follow human-centric design principles and leave meaningful opportunity for human choice.
- At the same time, proper oversight mechanisms need to be ensured, which can be achieved through human-in-the-loop, human-on-the-loop, and human-in-command approaches.

Technical Robustness and safety

- AI systems need to be resilient and secure.
- They need to be safe, ensuring a fall back plan in case something goes wrong.
- They need to be **accurate**, **reliable** and **reproducible**. That is the only way to ensure that also unintentional harm can be minimized and prevented.

Privacy and data governance

Besides ensuring full respect for **privacy and data protection**, adequate data governance mechanisms must also be ensured, taking into account the **quality and integrity of the data**, and ensuring legitimised access to data.

To allow individuals to trust the data gathering process, it must be ensured that data collected about them will not be used to unlawfully or unfairly discriminate against them.

Privacy and data governance

Quality and integrity of the data: When data is gathered, it may contain socially constructed biases, inaccuracies, errors and mistakes. This needs to be addressed prior to training with any given data set.

Access to data:

- Data protocols governing data access should be put in place.
- These protocols should outline who can access data and under which circumstances.
- Only duly qualified personnel with the competence and need to access individual's data should be allowed to do so.

Transparency

The data, system and AI business models should be transparent. Traceability mechanisms can help achieving this.

Moreover, AI systems and their decisions should be explained in a manner adapted to the stakeholder concerned. Humans need to be aware that they are interacting with an AI system and must be informed of the system’s capabilities and limitations.

Diversity, Non-discrimination and Fairness

- Unfair bias must be avoided, as it could have multiple negative implications, from the marginalization of vulnerable groups, to the exacerbation of prejudice and discrimination.

- Fostering diversity, AI systems should be accessible to all, regardless of any disability, and involve relevant stakeholders throughout their entire life circle.

Societal and Environmental well-being

AI systems should benefit all human beings, including future generations.

It must hence be ensured that they are **sustainable** and **environmentally friendly**.

Moreover, they should take into account the environment, including other living beings, and their social and societal impact should be carefully considered.

Accountability

- Mechanisms should be put in place to ensure responsibility and accountability for AI systems and their outcomes.
- Auditability, which enables the assessment of algorithms, data and design processes plays a key role therein, especially in critical applications.
- Further, adequate and accessible redress should be ensured.

Limitations

- The Limitations of Machine Learning

https://towardsdatascience.com/the-limitations-of-machine-learning-a00e0c3040c6

- The Limitations of Deep Learning

https://blog.keras.io/the-limitations-of-deep-learning.html

- The Future of AI; Bias Amplification & Algorithmic Determinism

https://digileaders.com/future-ai-bias-amplification-algorithmic-determinism/

What Makes Tiny Machine Learning

What is Tiny Machine Learning (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**, typically in the mW range and below, and hence enable a variety of always-on ML use-case on battery-operated devices.

TinyML & its Usage

On-device machine learning applications in the single mW and below

Vibration and motion

Any ‘signal’

Predictive maintenance, sensor fusion, accelerometer, pressure, lidar/radar, speed, shock, vibration, pollution, density, viscosity, etc.

Voice and sound

Recognition and creation

Keyword spotting, speech recognition, natural language processing, speech synthesis, sound recognition, etc.

Vision

Images and video

Object detection, face unlock, object classification etc.

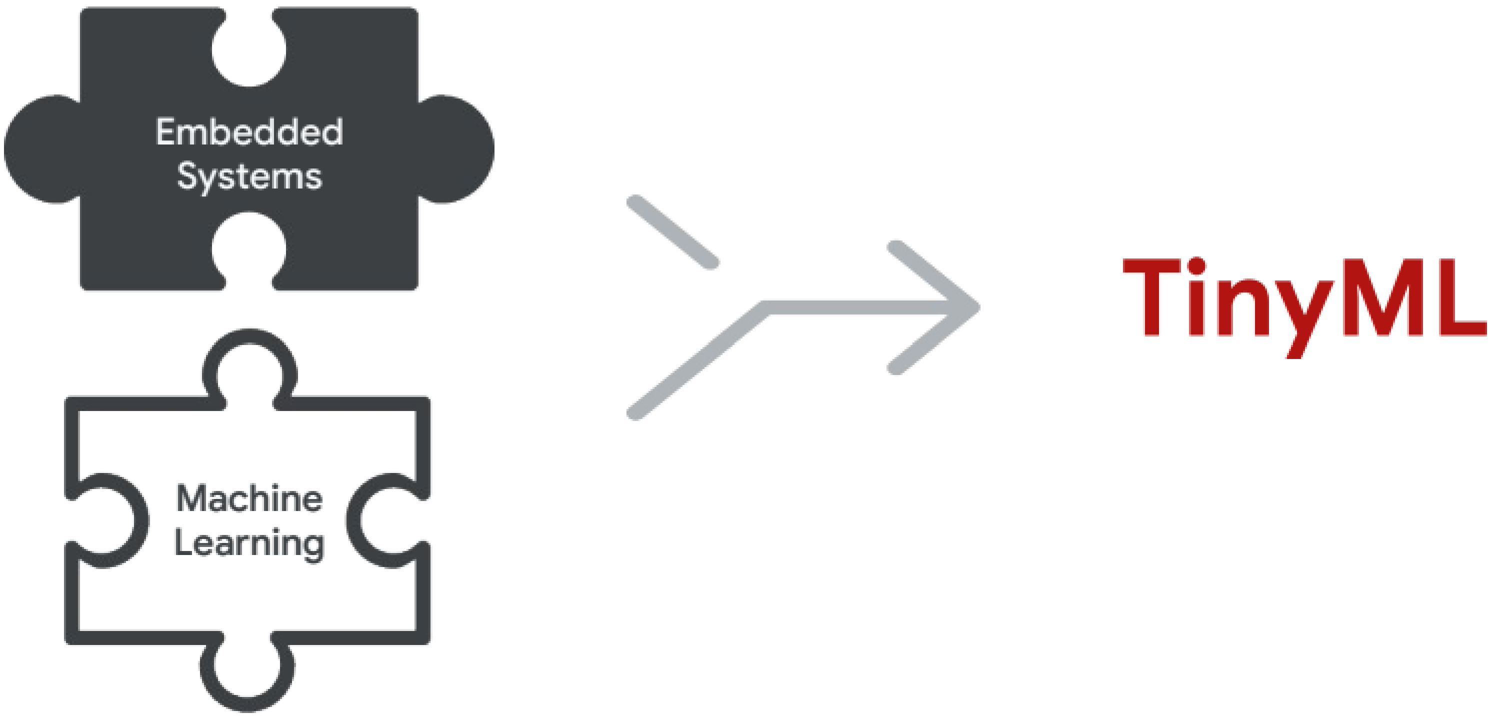
arm

Apple Watch (Example of Battery driven ML)

Intelligent Drones

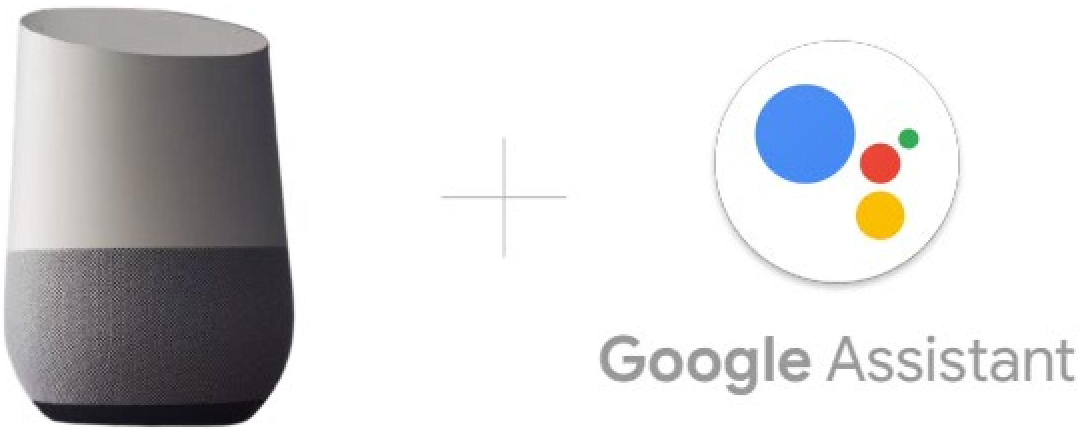
https://www.youtube.com/watch?v=E0zGIWHYV3Q

What makes Tiny Machine Learning

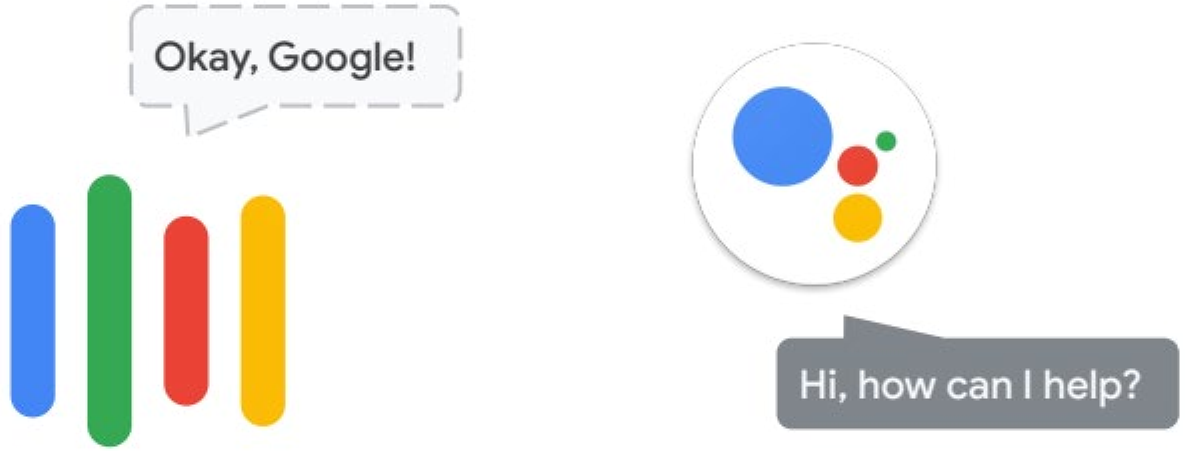


Execute the machine learning at the tiny **endpoint devices** rather than in the powerful general computer or cloud platform.

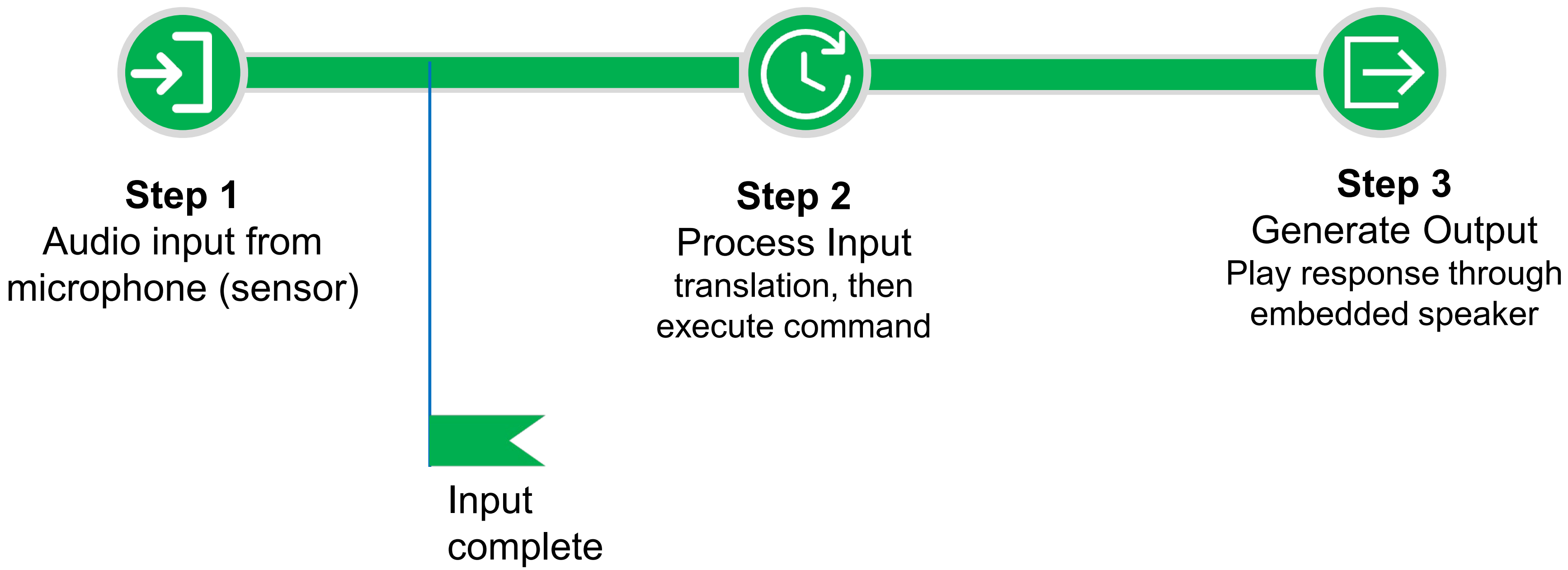
Let’s Take an Example



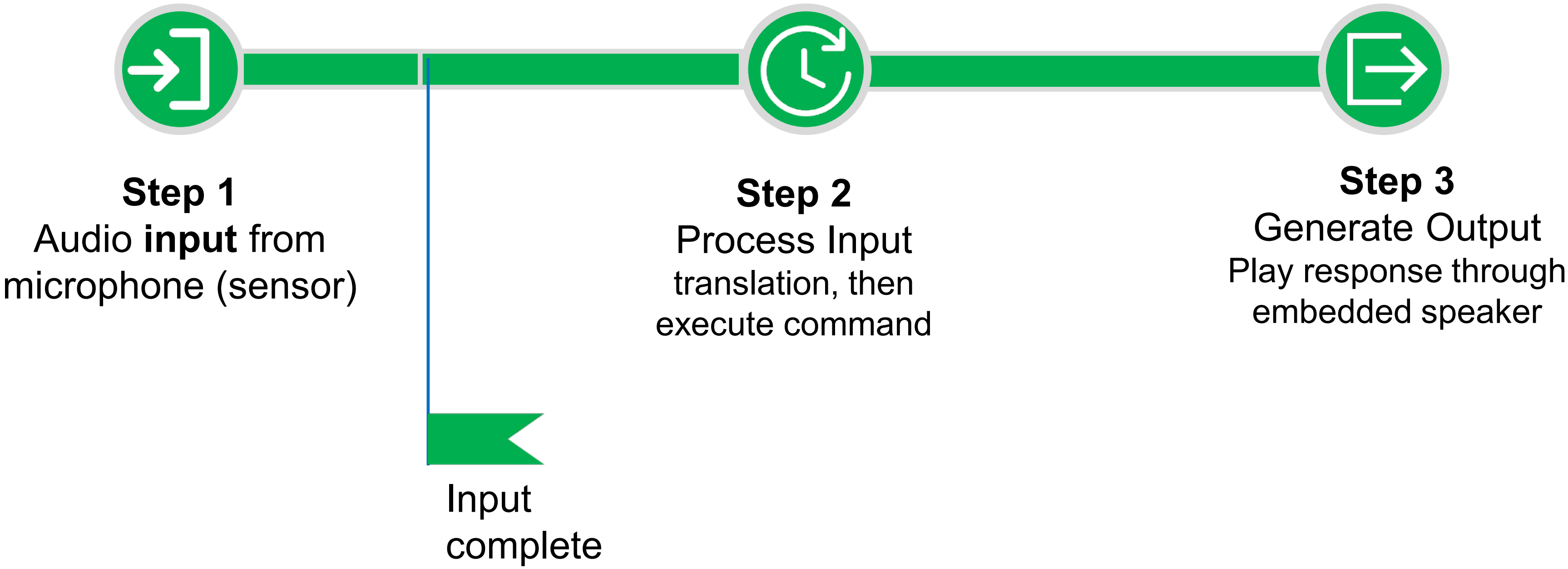
Let’s Take an Example



The Three Basic Steps



Input


Step 1
Audio **input** from
microphone (sensor)
Step 2
Process Input
translation, then
execute command
Step 3
Generate Output
Play response through
embedded speaker
Input
complete

Endpoints Have Sensors, Tons of Sensors

Motion Sensors
Gyroscope, Radar, Magnetometer, Accelerator

Acoustic Sensors
Ultrasonic, Microphones, Geophones, Vibrometers

Environmental Sensors
Temperature, Humidity, Pressure, IR, etc.

Touchscreen Sensors
Capacitive, IR

Image Sensors
Thermal, Image

Biometric Sensors
Fingerprint, Heart rate, etc.

Force Sensors
Pressure, Strain

Rotation Sensors
Encoder

Other Sensors
Flow, Color, Radiation, Voltage & Current, Gas

Biometric Sensors

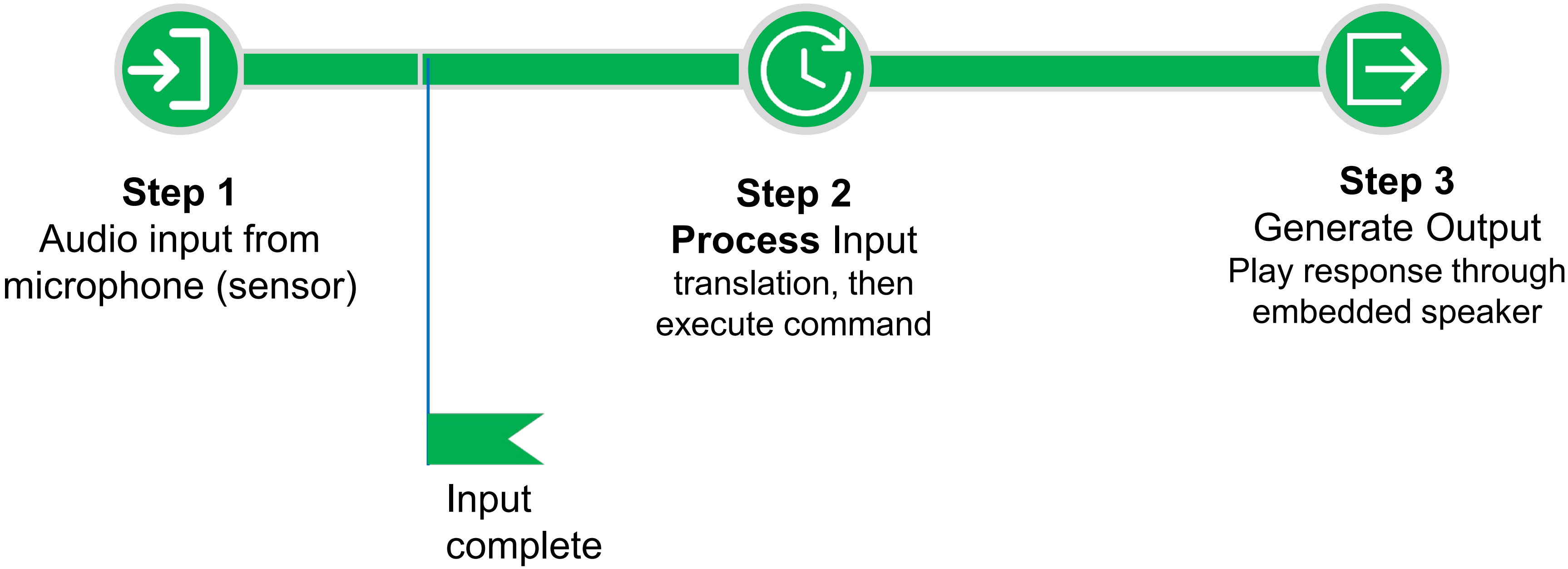
Non-invasive Glucose Monitoring

Fingerprint + Photoplethysmography (**PPG**)

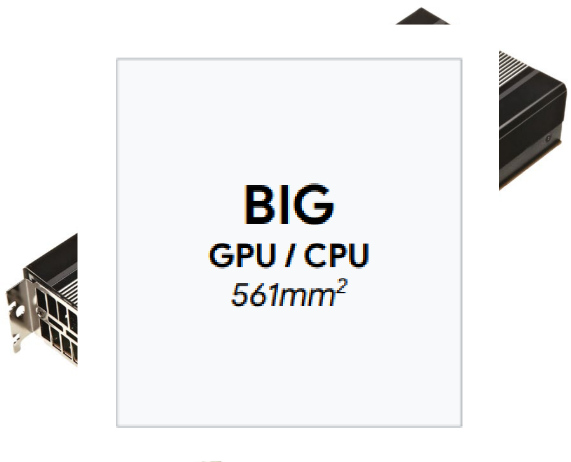
PPG is a non-invasive technology that uses a light source and a photodetector at the surface of skin to measure the volumetric variations of blood circulation.

Source: Jacobs School of Engineering/UC San Diego

Processing

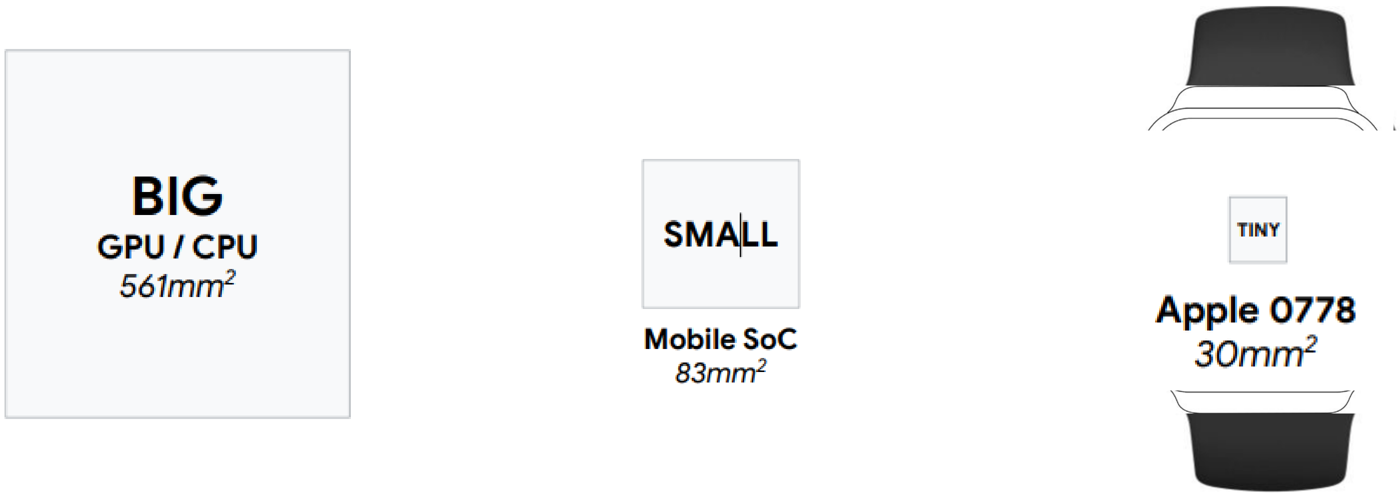

Step 1
Audio input from microphone (sensor)
Step 2
Process Input
translation, then execute command
Step 3
Generate Output
Play response through embedded speaker
Input complete

Thinking Big



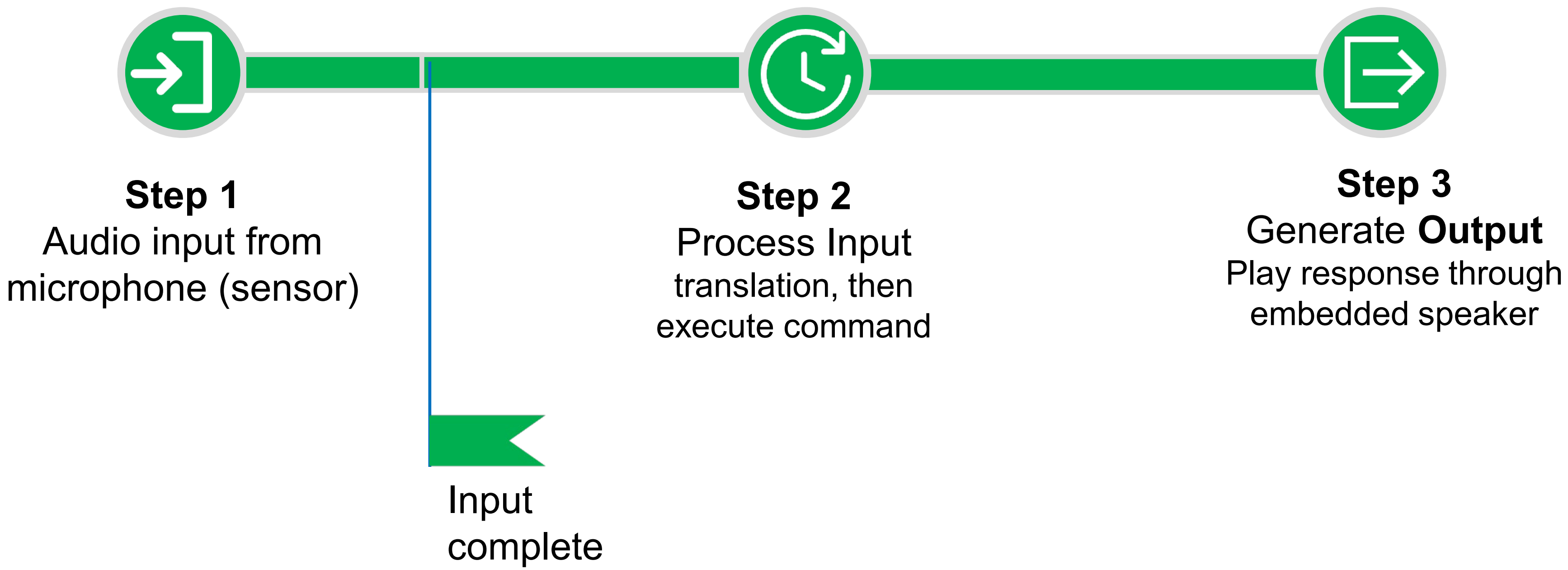
Thinking Small

Thinking **Tiny**



Thinking Record-breaking

Output


Step 1
Audio input from microphone (sensor)
Input complete
Step 2
Process Input
translation, then execute command
Step 3
Generate **Output**
Play response through embedded speaker

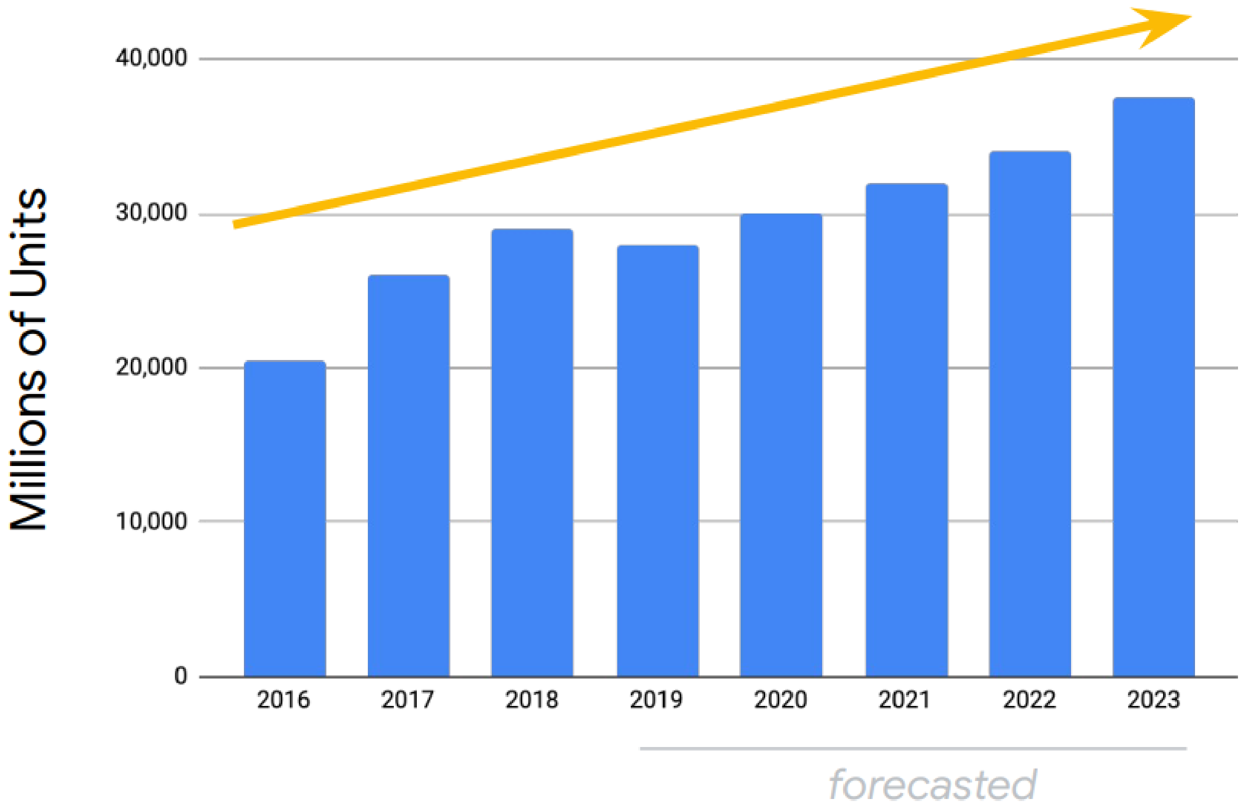
Application Determines Output Forms

Why TinyML is important

Tiny Computer Are Ubiquitous

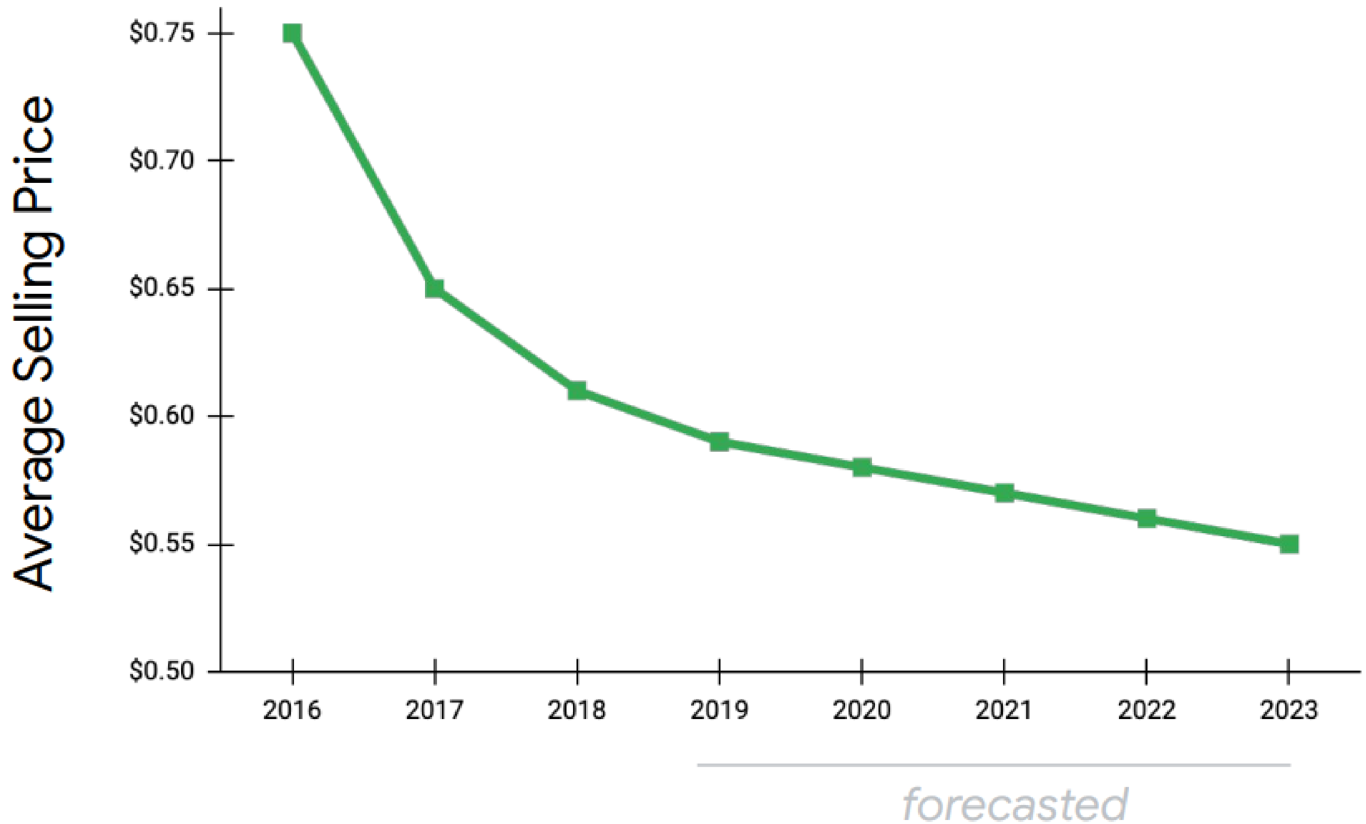
More than **250 Billion** Microcontrollers (MCUs) today

MCU Demand Forecast



Source: IC Insights

MCUs are Cheap



Source: IC Insights

MCUs are Ultra-low Power System

Use case: button cell battery

Neural Decision Processor

Always-on deep learning speech/audio recognition

Ultra low power, 128KB SRAM, 12-pin, 2.52mm^2

140 µW

Syntiant NDP100

No Good Data Left Behind

5 Quintillion

bytes of data produced every day by IoT

Quintillion 10^{18} 10^{30}

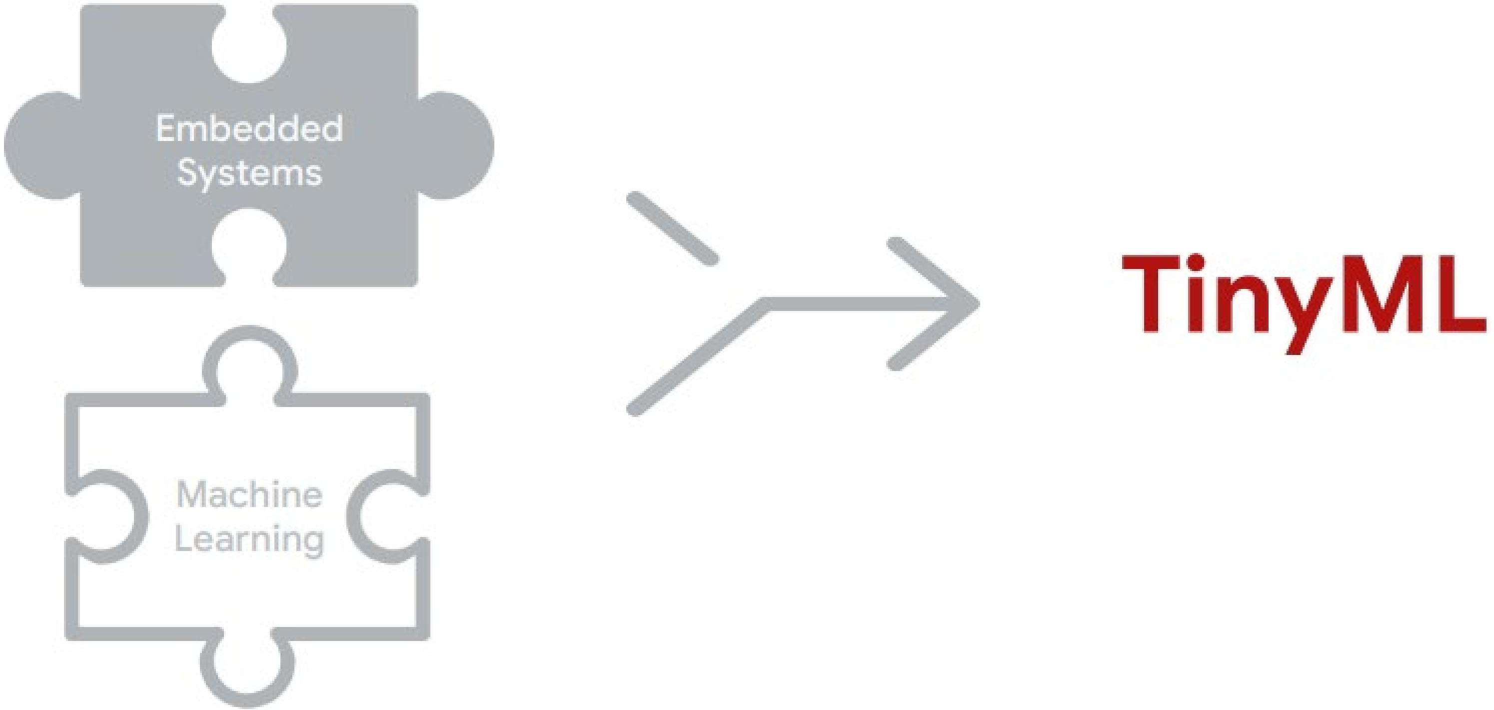
< 1%

of unstructured data is analysed or used at all

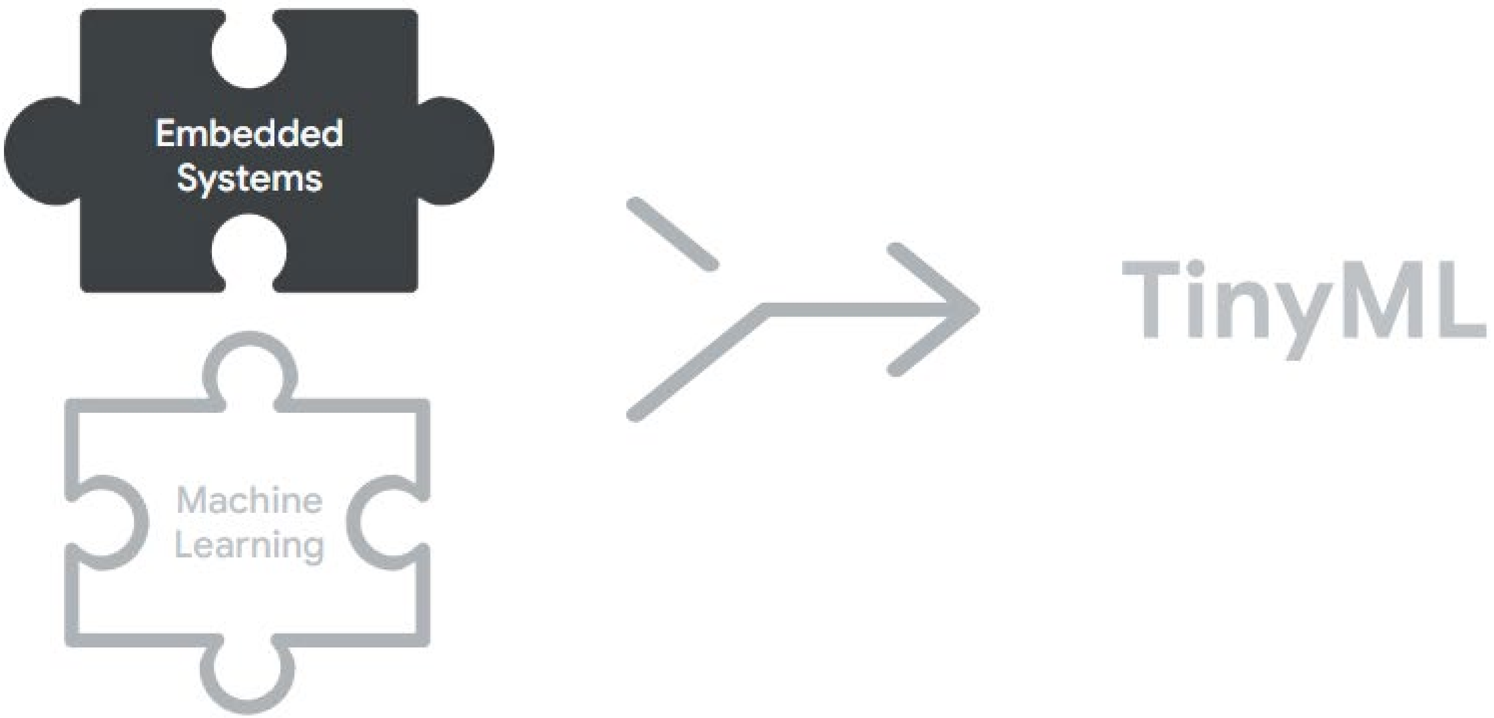
Source: Harvard Business Review, What's Your Data Strategy?, April 18, 2017
Cisco, Internet of Things (IoT) Data Continues to Explode Exponentially. Who Is Using That Data and How?, Feb 5, 2018

The Challenges of TinyML

Challenges from the Embedded Systems



Challenges from the Embedded Systems



Hardware

Compute

Memory

Storage

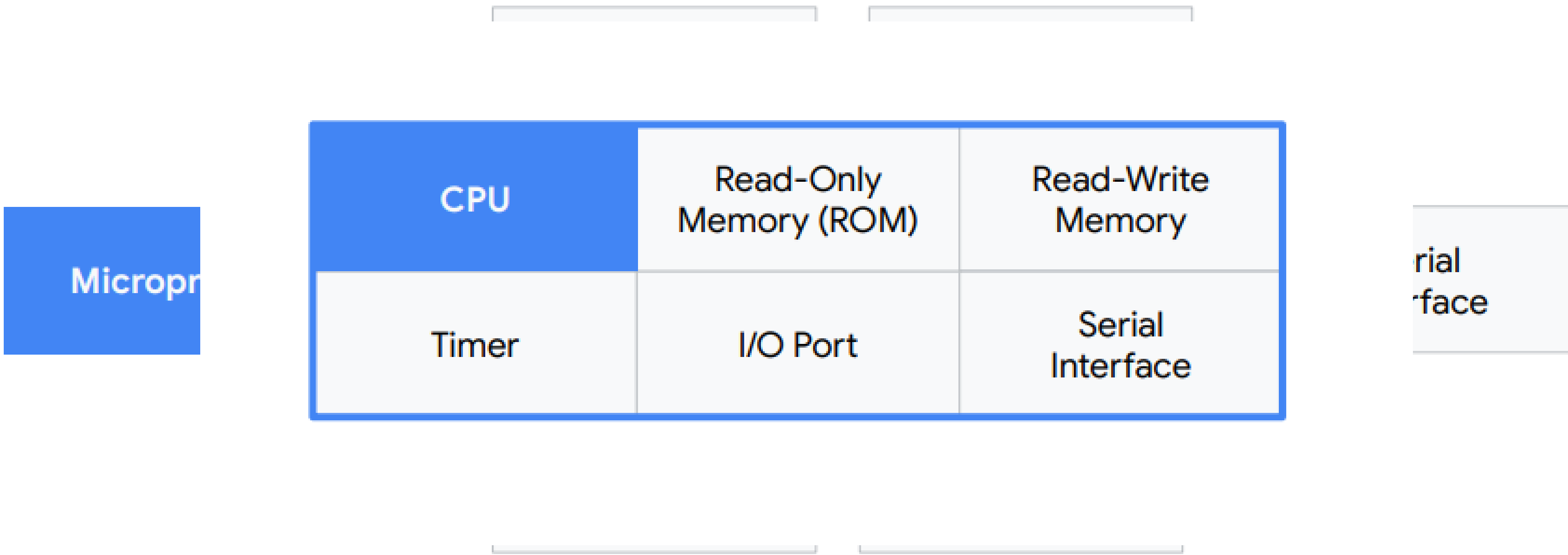
Differences?

Microprocessor

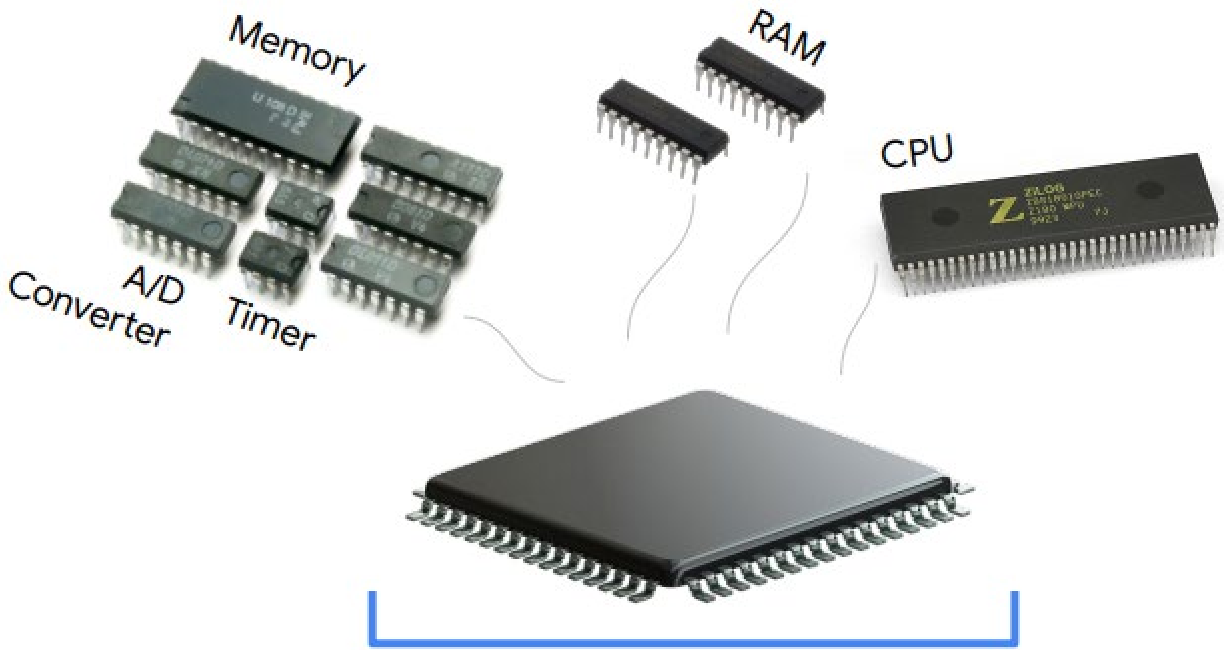
v.s.

Microcontroller

Microprocessor



Microcontroller: a complete package



Microprocessor V.S. Microcontroller

THE UNIVERSITY OF WESTERN AUSTRALIA

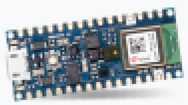
Microprocessor

- Heart of a **computer system**
- Just the processor, memory and storage are **external**
- Mainly used in **general purpose systems** like laptops, desktops and servers
- **Offers flexibility** in design
- System size is **big**

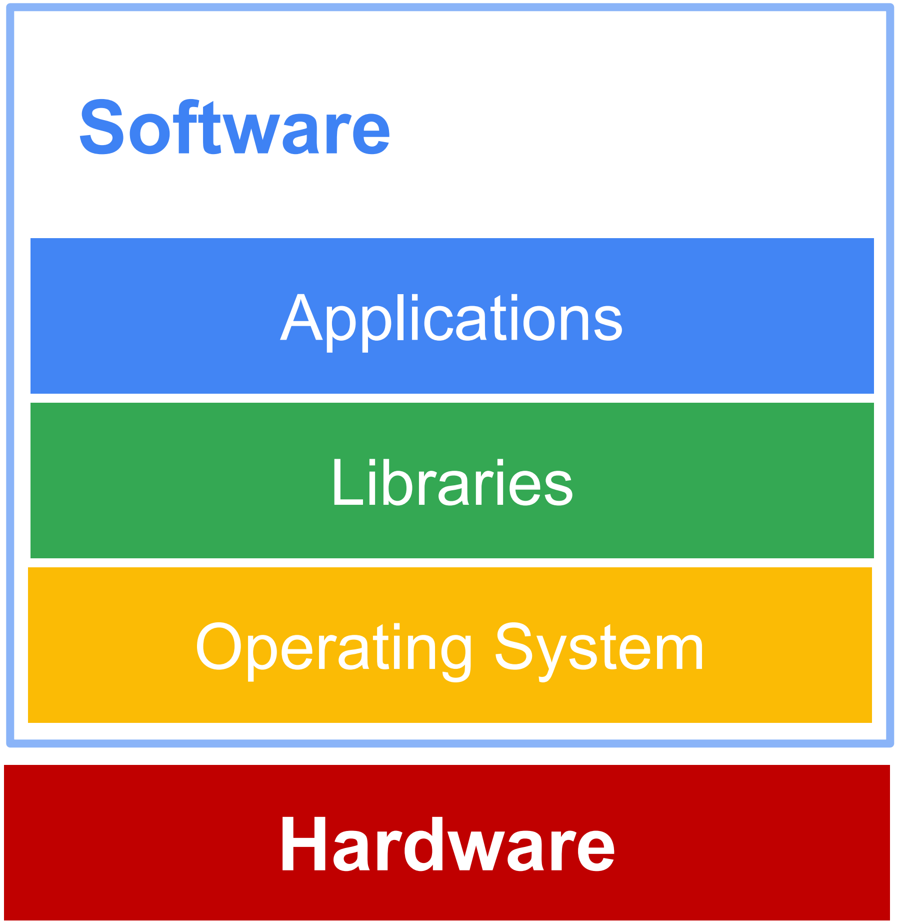
Microcontroller

- Heart of an **embedding system**
- Memory and storage are all **internal** to the system
- Mainly used in **specialized, fixed function systems** like phones, MP3 players, etc.
- **Limited flexibility** in design
- System size is **tiny**

Order of Magnitude Difference

	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

Software


Software
Applications
Libraries
Operating System
Hardware

Widely Used Operating Systems

Widely Used Operating Systems

Embedded Sys.

?

Mbed is a development platform and operating system fo
devices based on 32-bit ARM Cortex-M microcontrollers.

Real Time Operating System

- A real-time operating system (RTOS) is an operating system (OS) for real-time computing applications that processes data and events that have critically defined time constraints.

- An RTOS is distinct from a time-sharing operating system that manages the sharing of system resources with a scheduler, data buffers, or fixed task prioritization in a multitasking or multiprogramming environments.

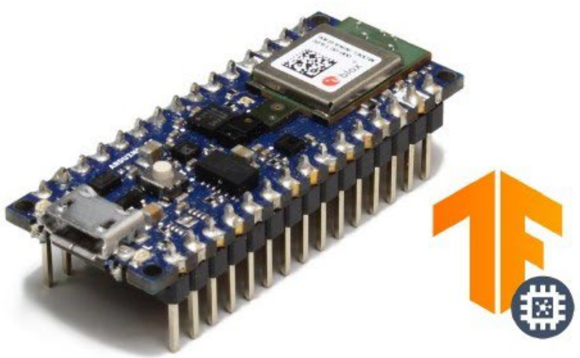
Real Time Operating System

- All processing must occur within the defined constraints.
- Event-driven systems switch between tasks based on their priorities, while time-sharing systems switch the task based on clock interrupts

Embedded Sstem

Single Board Computer

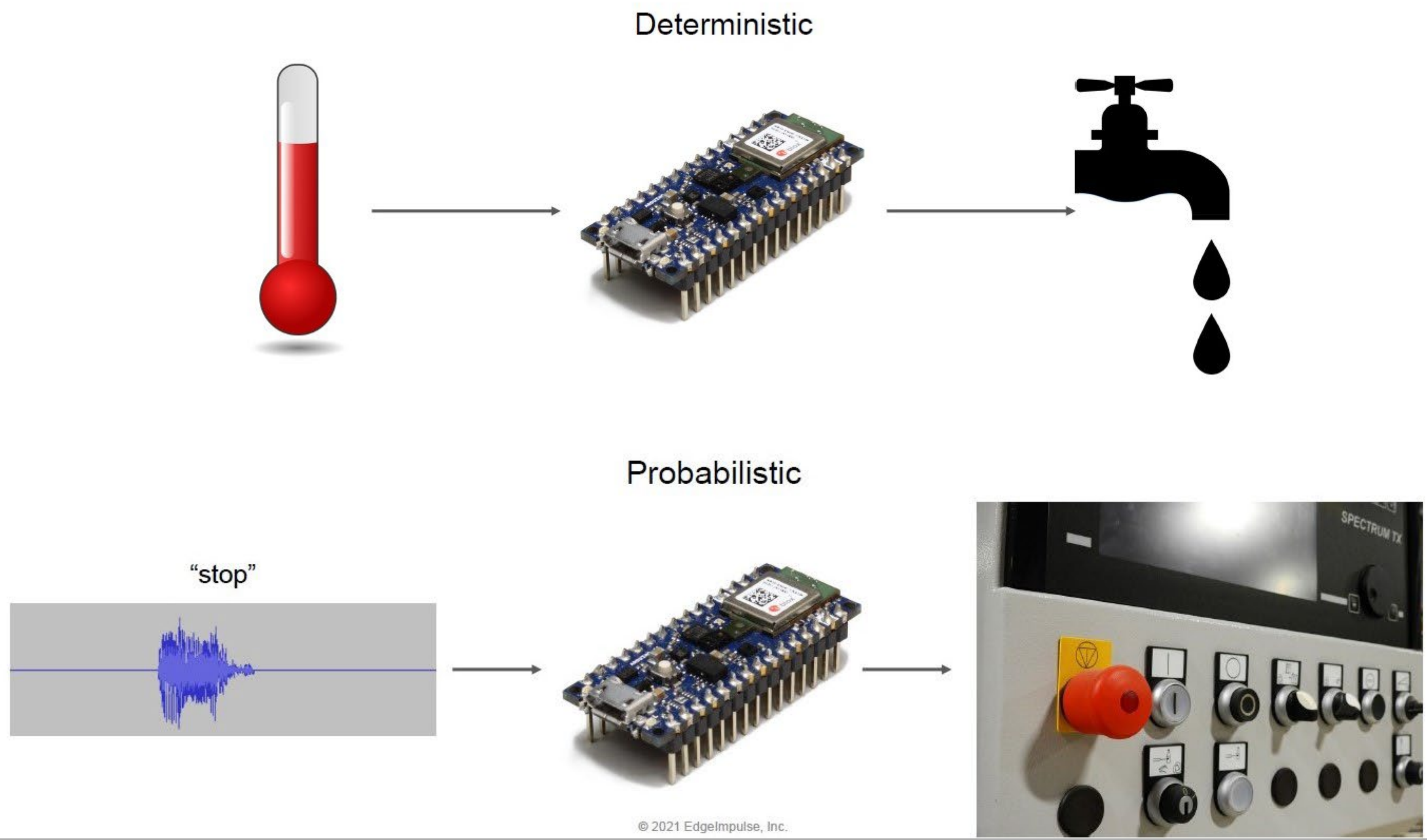
- More powerful (faster processor, more memory)
- Runs full, general purpose operating system (OS)
- Can provide full command line or graphical user interface
- Requires more power



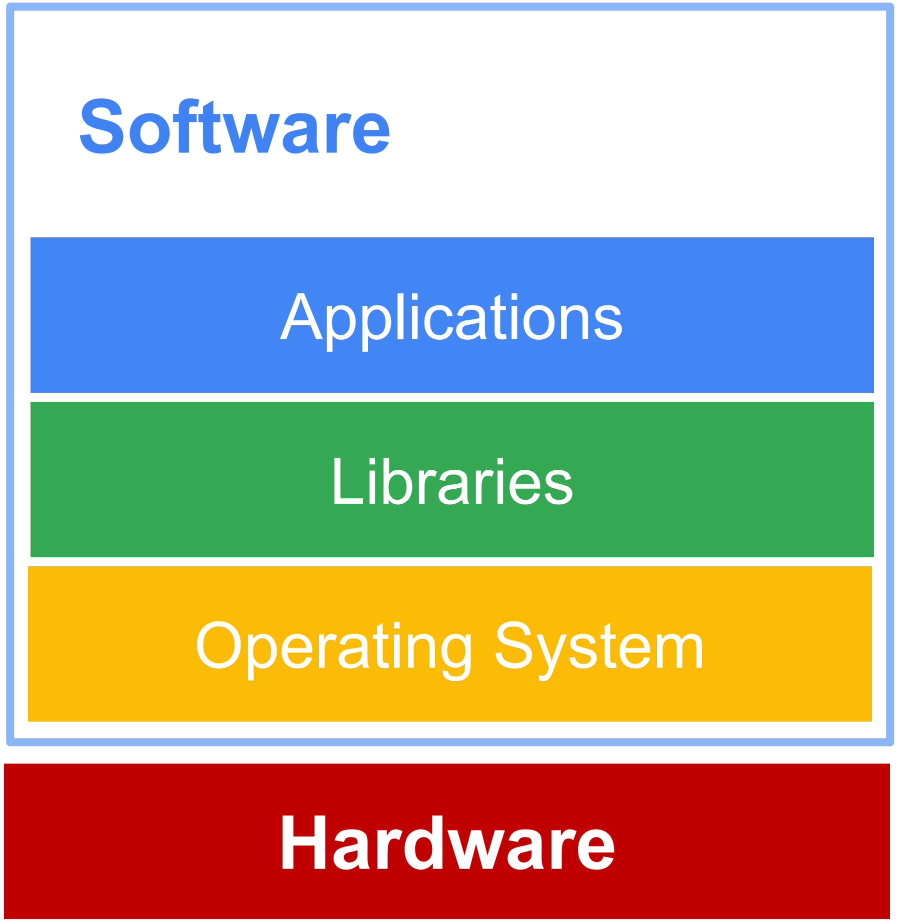
Microcontroller

- Less powerful
- Bare-metal (superloop) or real-time operating system (RTOS)
- Limited or no user interface
- Requires less power

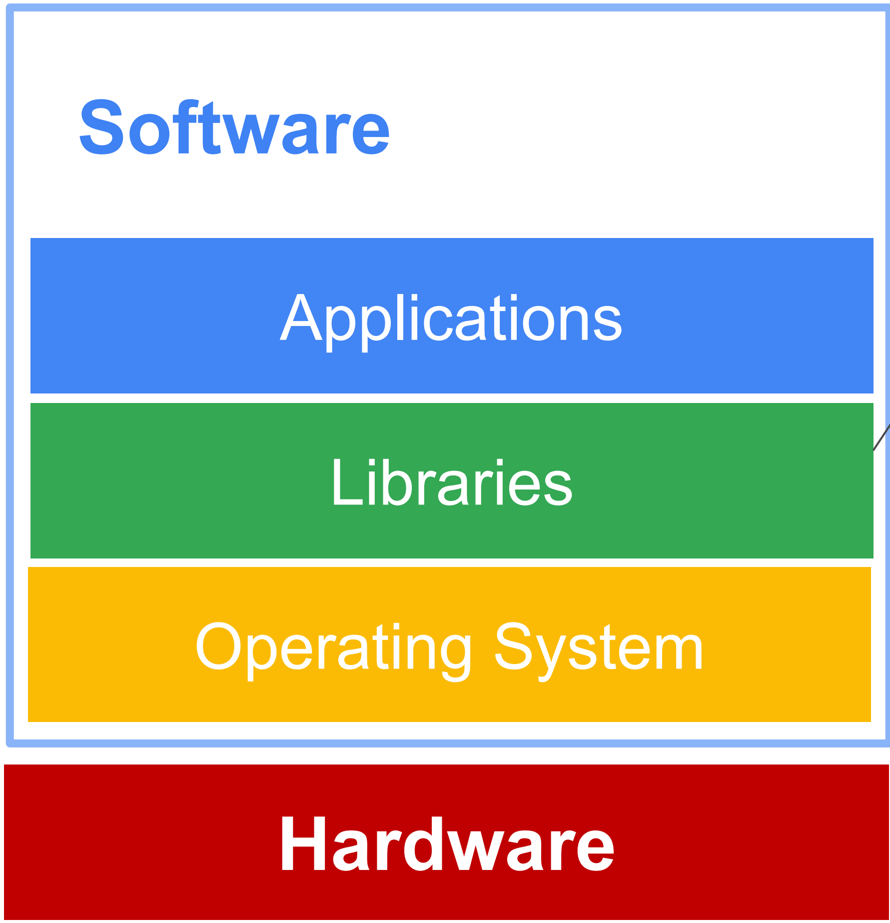
Micro-controller Usage



Libraries


Software
Applications
Libraries
Operating System
Hardware

Libraries




```
import numpy as np
```

Libraries

Software

Applications

Libraries

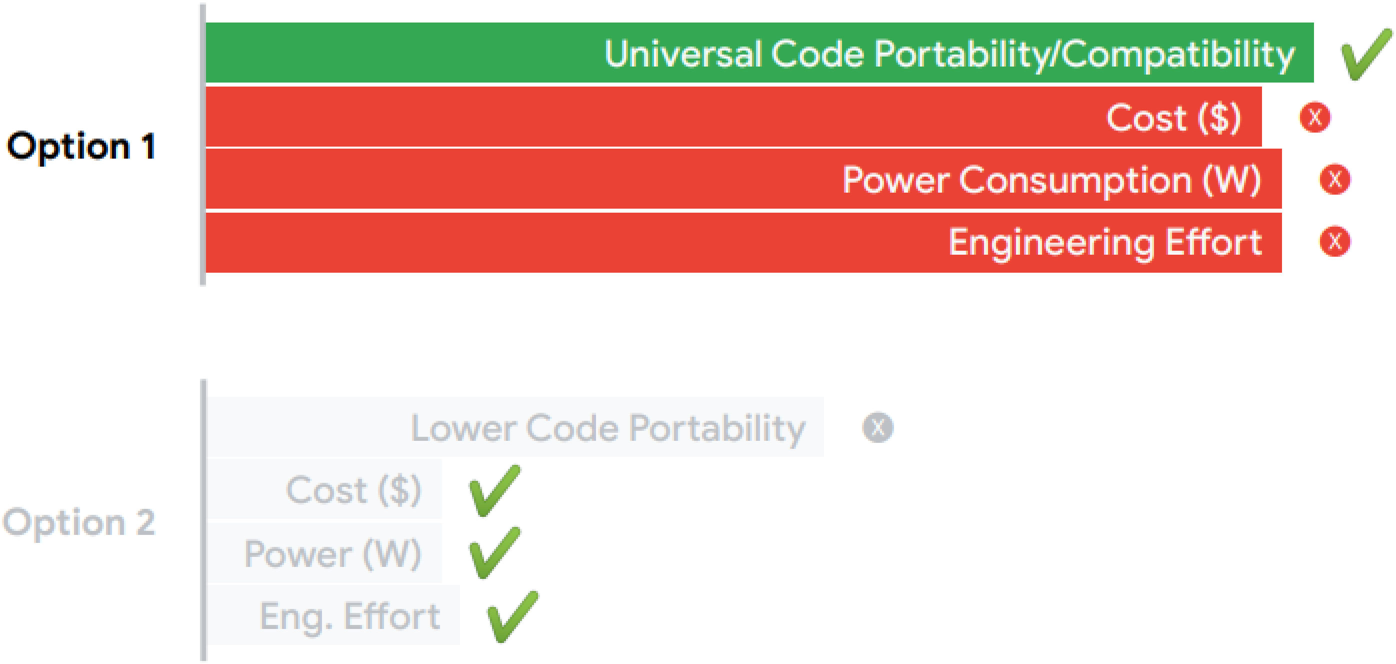
Operating System

Hardware

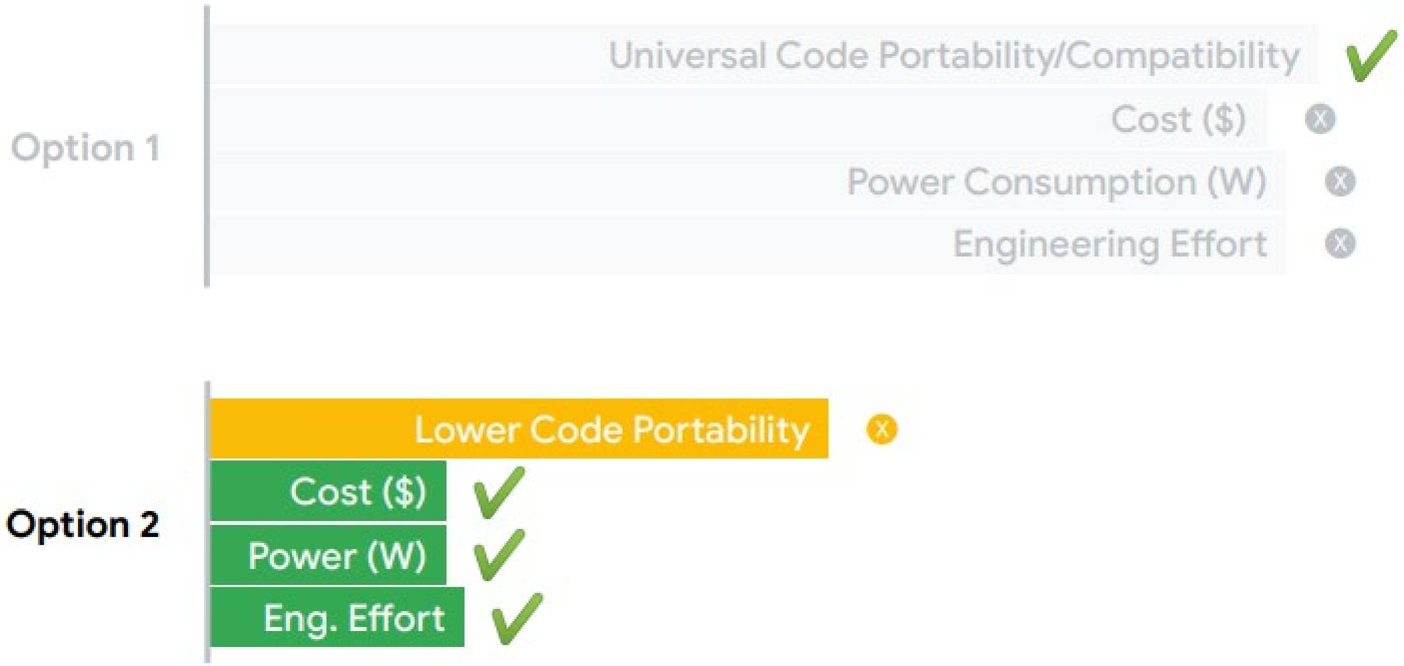
Portability Opportunity

Able to execute the same code on different microprocessor hardware and architectures.

Portability Trade-offs



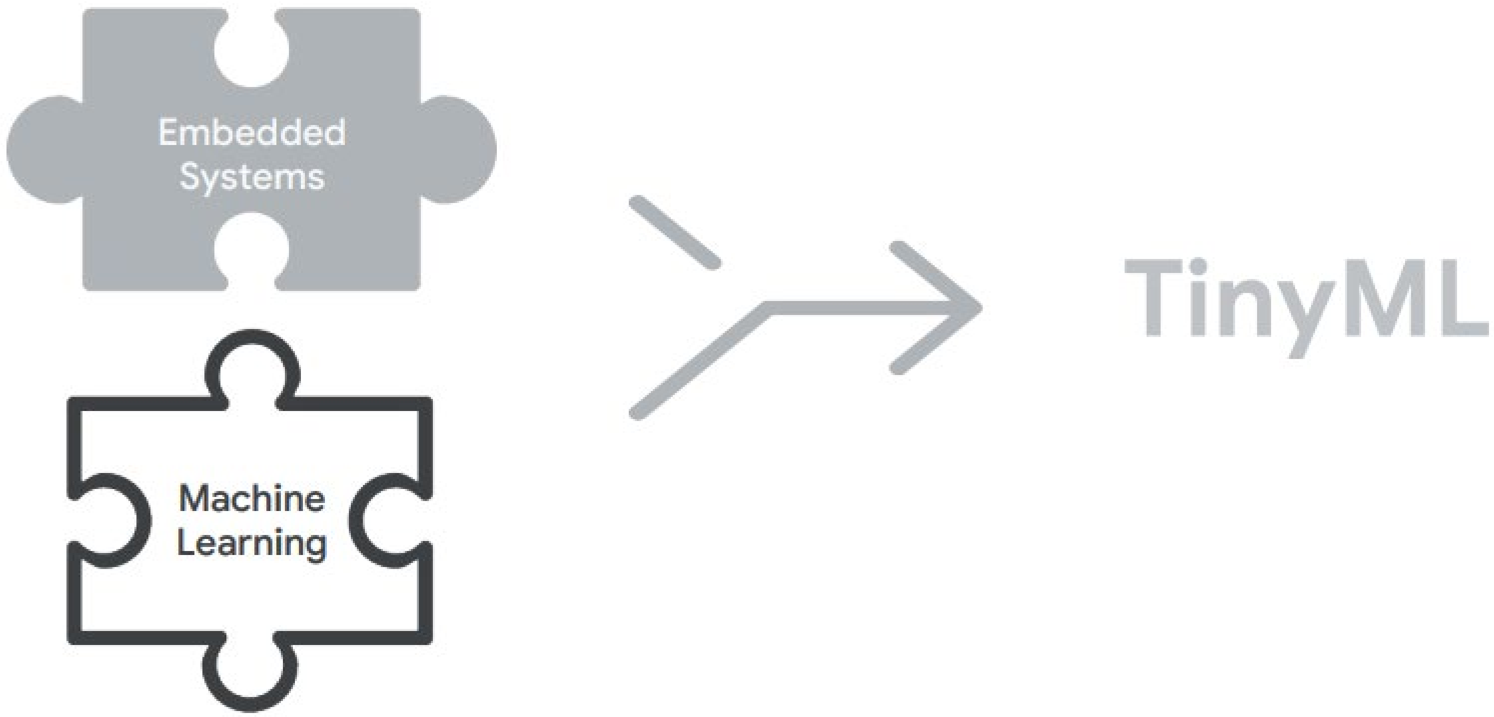
Portability Trade-offs



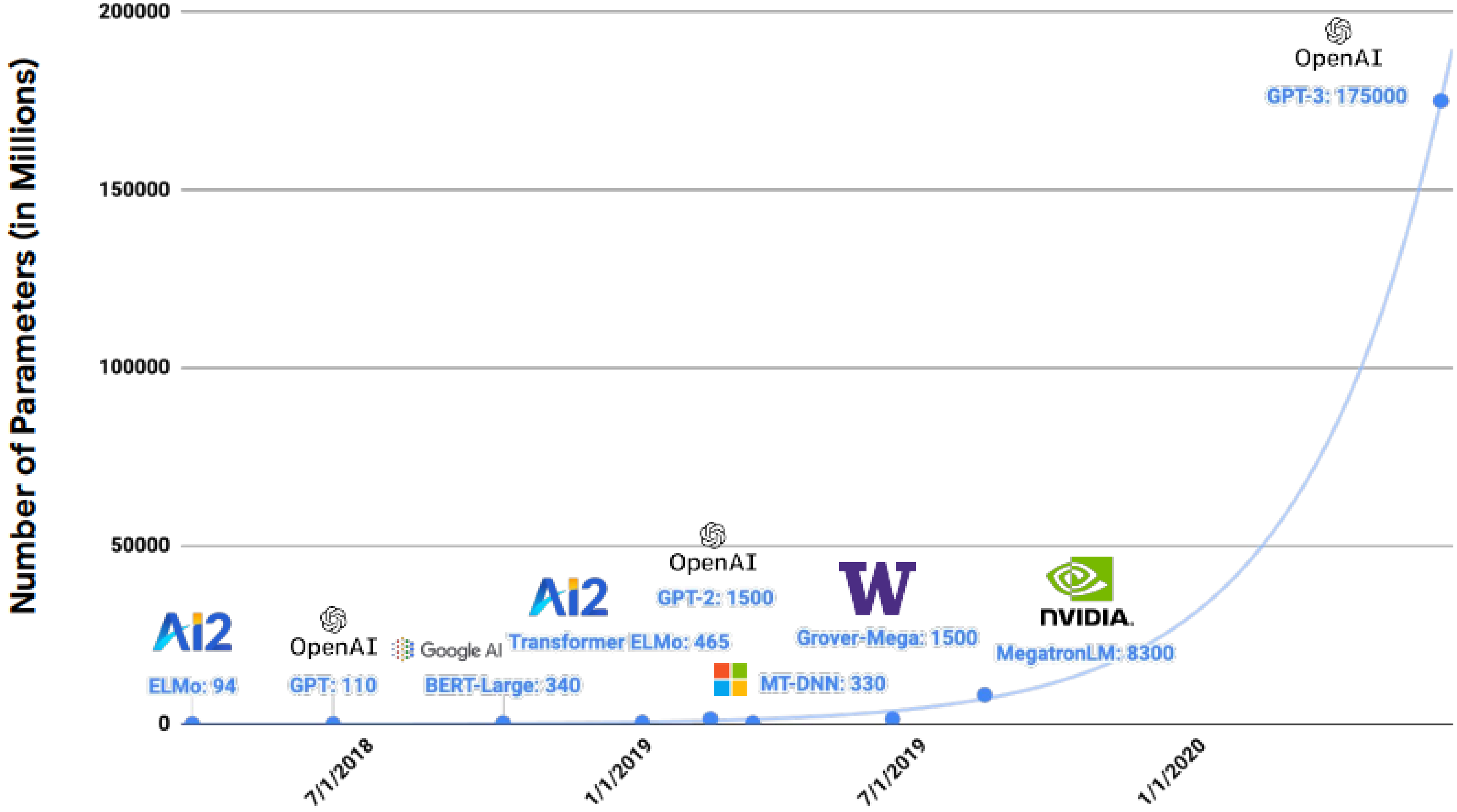
Summary

- **Embedded hardware** is extremely limited in performance, power consumption and storage
- **Embedded software** is not as portable and flexible as mainstream computing

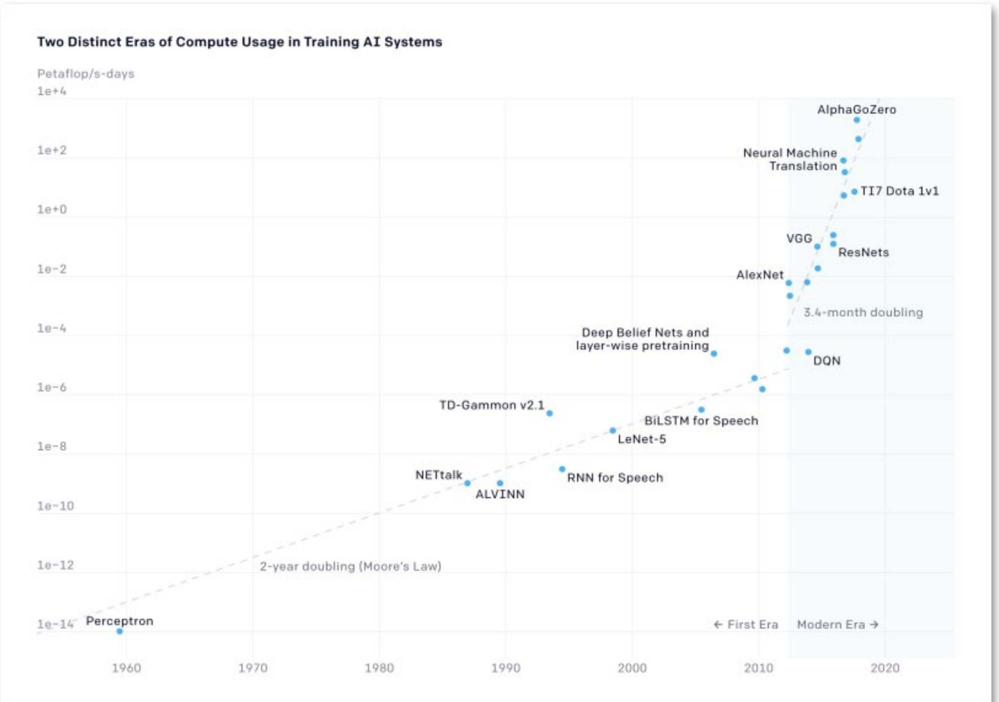
Challenges from the Machine Learning



ML Model Size Growth



ML Compute Needs (from the 1960s)



Source: https://openai.com

In recent years, the amount of computing needed has grown remarkably fast.

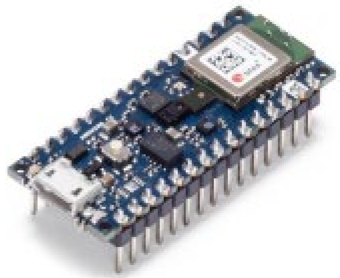
Compute requirements are **doubling nearly every 3 to 4 months**

Petaflop : A measure of computing speed equal to one quadrillion floating-point operations per second. A petaflop/s-day (pfs-day) consists of performing 1015 neural net operations per second for one day, or a total of about 1020 operations.

From GPU/TPU to Embedded MCU

Cloud TPU

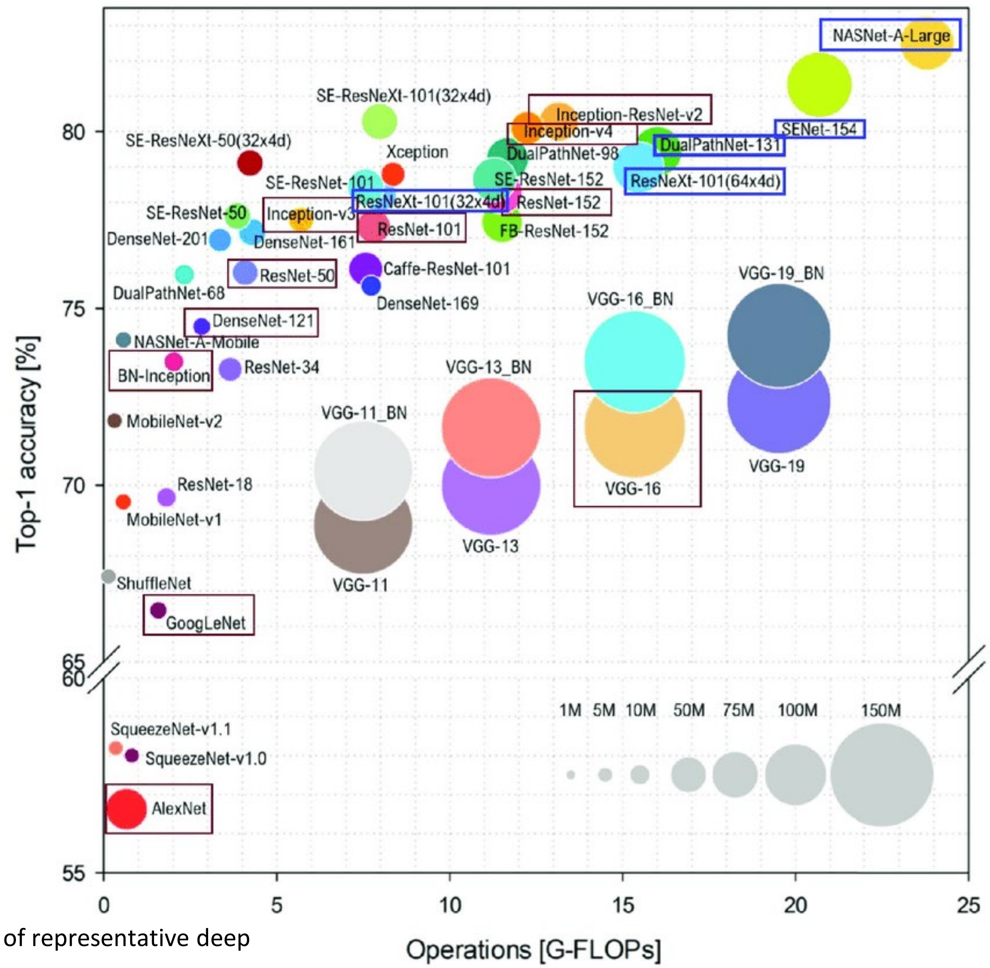
TinyML



Google Cloud Tensor Processing Units, TPUs are custom-designed AI accelerators, which are optimized for training and inference of large AI models.

ML Model Evolution

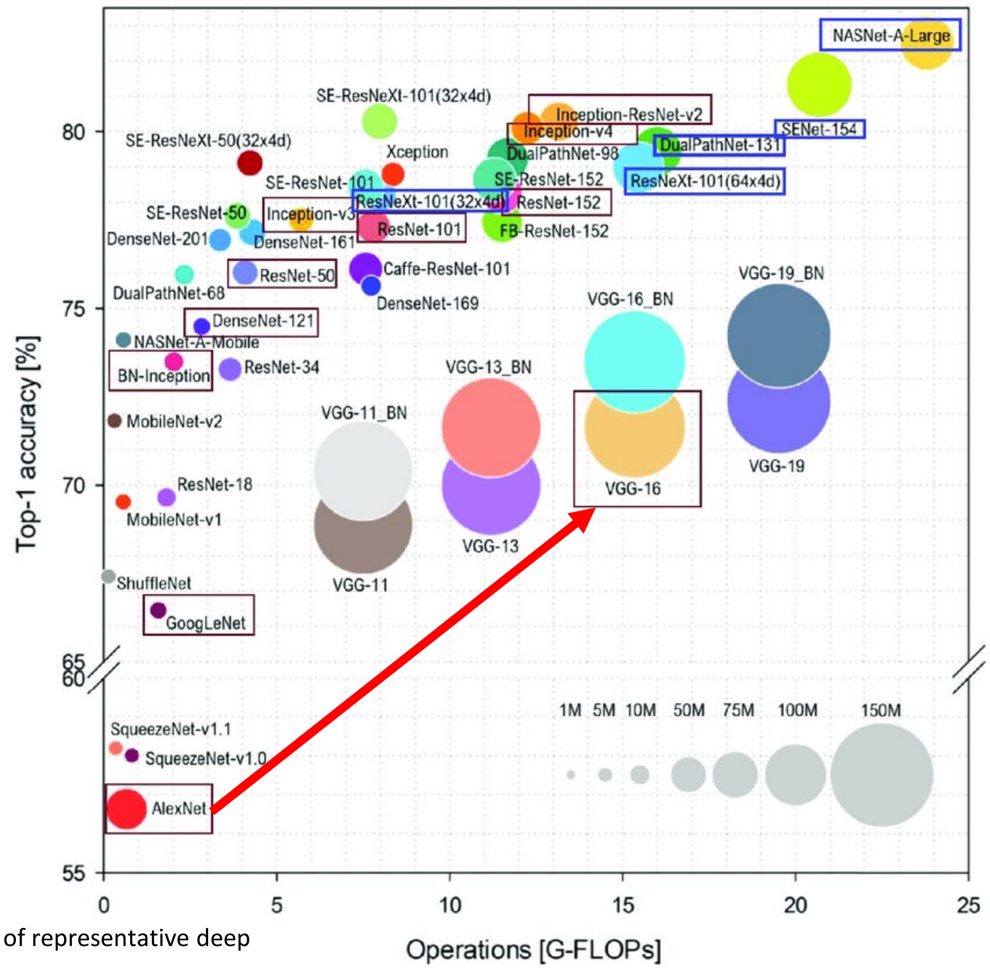
- **AlexNet (2012)**
  - 57.1% accuracy
  - 61MB in size



Source: S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018

ML Model Evolution

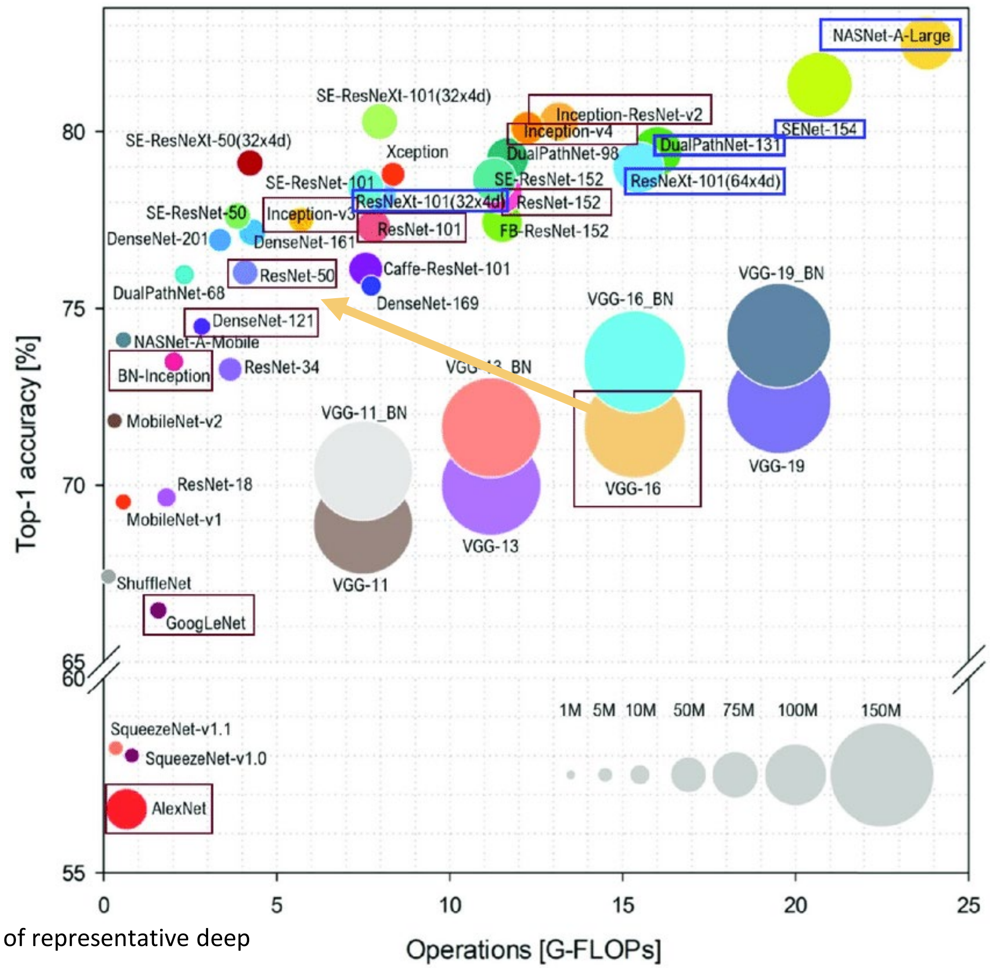
- **VGGNet(2014)**
  - 71.5% accuracy
  - 528MB in size



Source: S. Bianco, R. Cadene, L. Celona, and P. Napoletano, “Benchmark analysis of representative deep neural network architectures,” *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018

ML Model Evolution

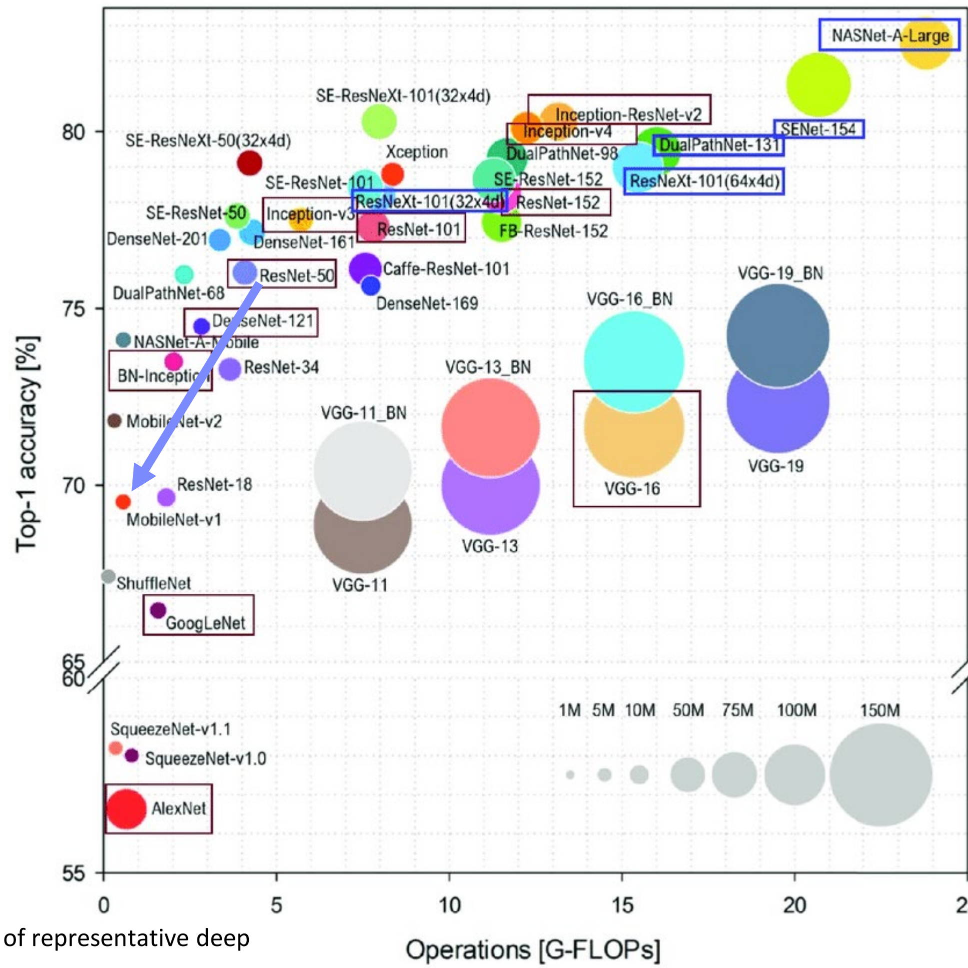
- **ResNet(2015)**
  - 75.8% accuracy
  - 22.7MB in size



Source: S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018

ML Model Evolution

- **MobileNet(2015)**
  - MobileNet-v1
    - 70.6% accuracy
    - 16.9MB in size



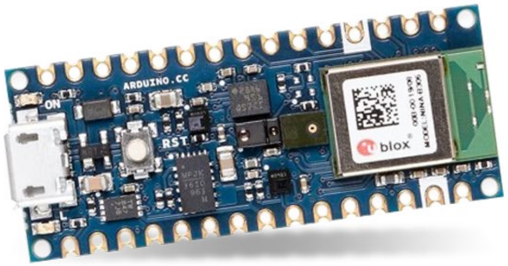
Source: S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018

Challenges of Model Deployment

- **MobileNet(2015)**
  - MobileNet-v1
    - 70.6% accuracy
    - 16.9MB in size

- **Problem**

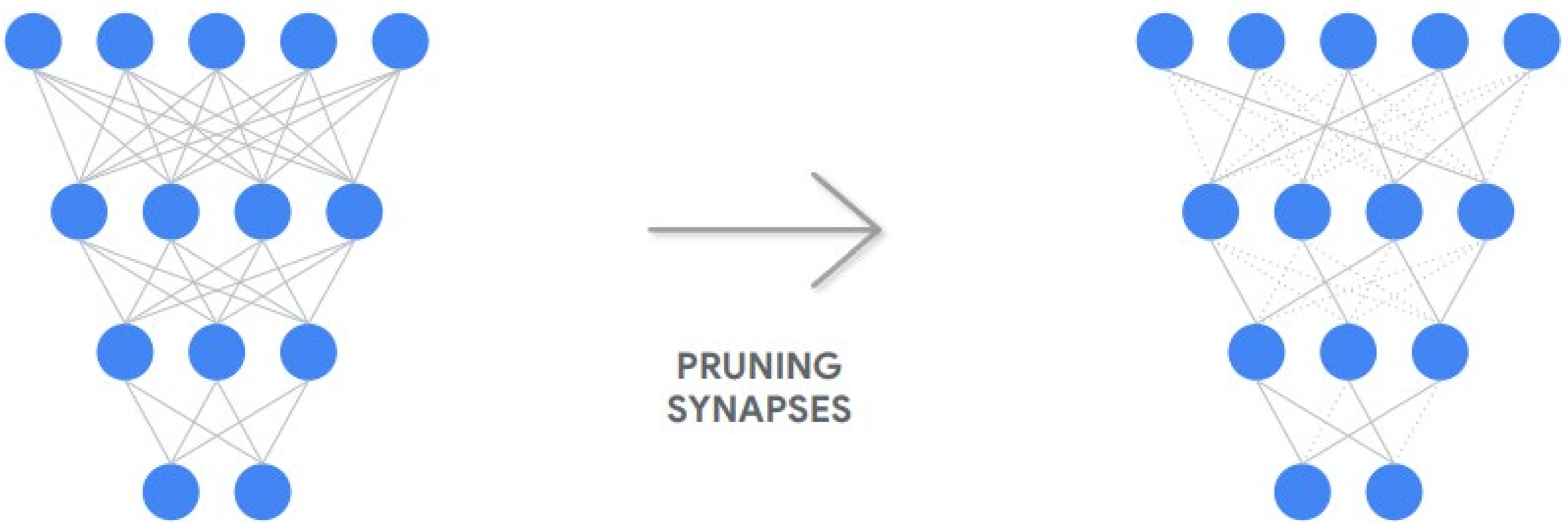
The board (Arduino Nano 33) only has **256KB** of RAM (memory) yet MobileNetv1 needs **16.9MB**.



Source: S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018

Model Compression Techniques

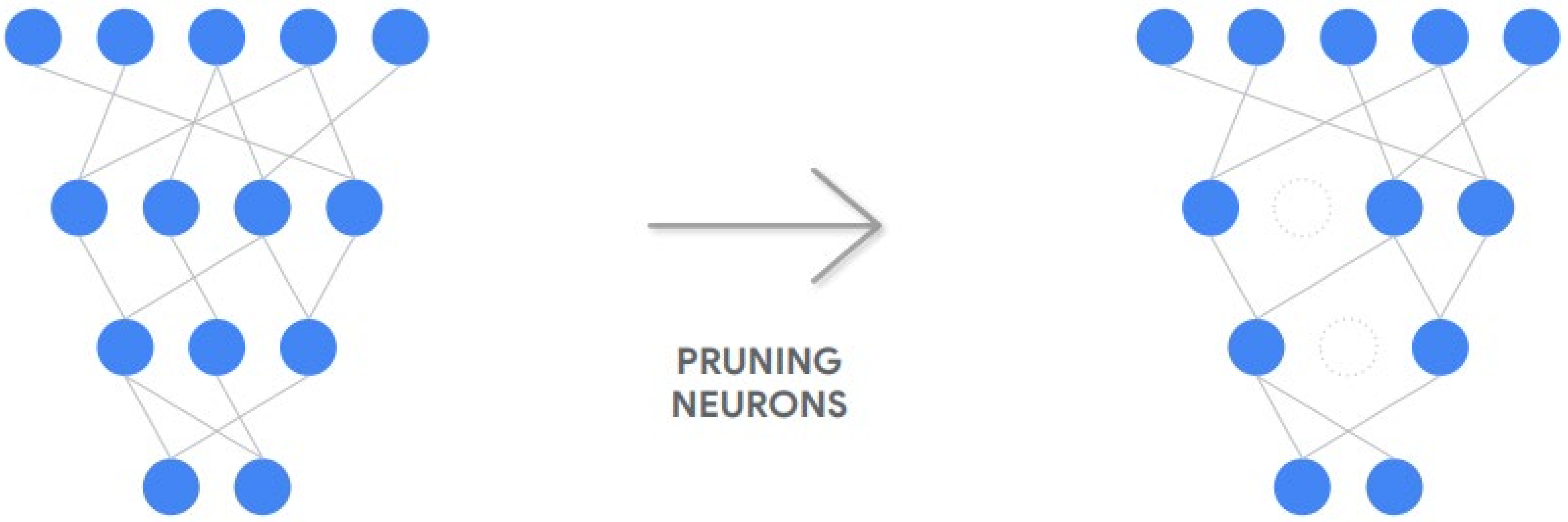
- **Pruning**



Pruning reduces the complexity of a neural network by identifying and eliminating weights or neurons that contribute little to the model's final predictions. This process leads to smaller, faster models with reduced memory and computational requirements

Model Compression Techniques

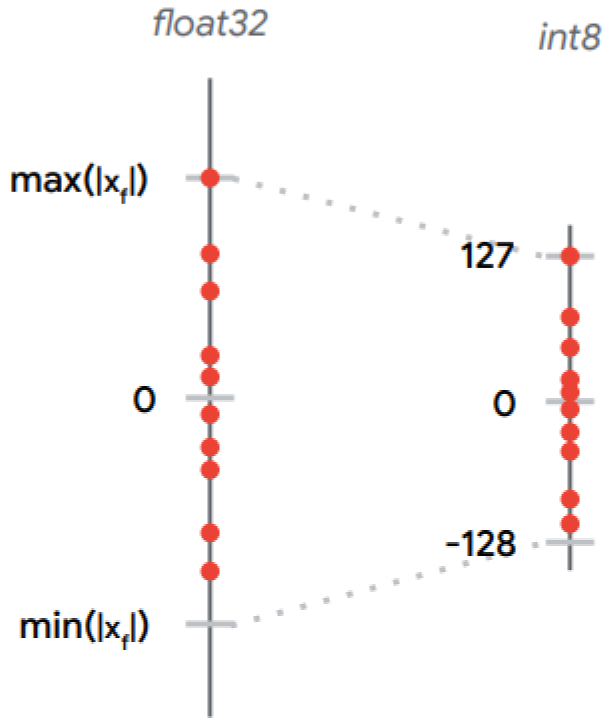
- **Pruning**



Pruning reduces the complexity of a neural network by identifying and eliminating weights or neurons that contribute little to the model's final predictions. This process leads to smaller, faster models with reduced memory and computational requirements.

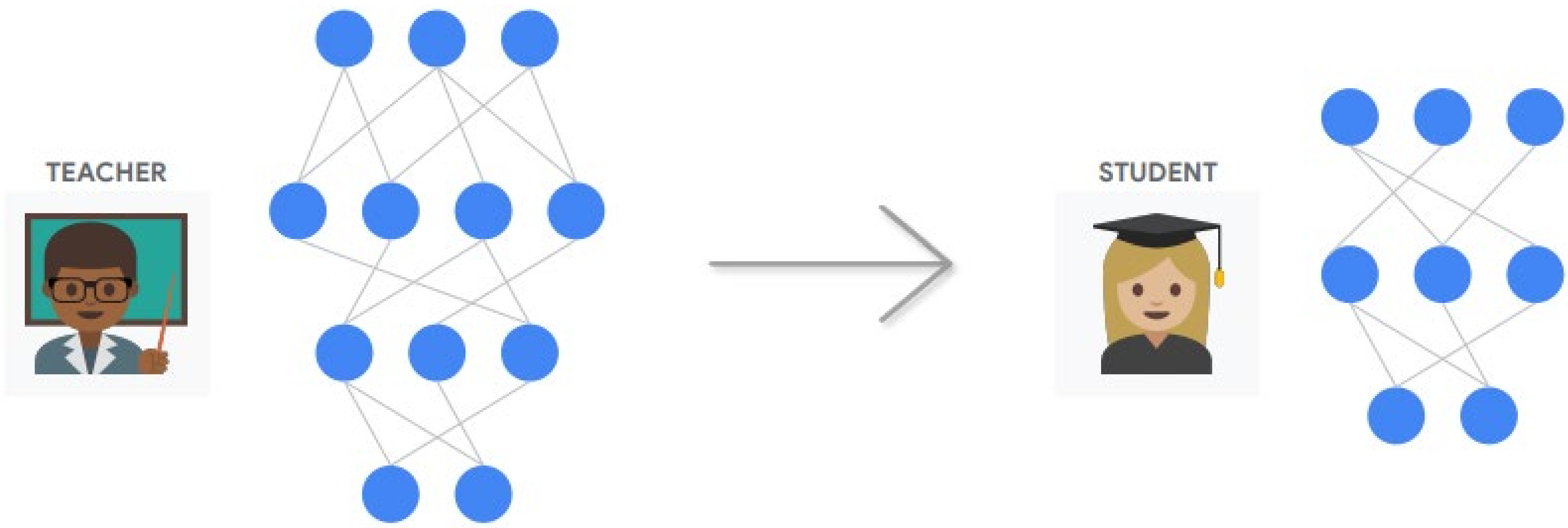
Model Compression Techniques

- **Quantization**



Model Compression Techniques

- **Knowledge Distillation**



The main goal of knowledge distillation is to transfer the knowledge learned by a large model into a smaller model, enabling faster inference and reduced memory requirements while maintaining comparable performance.

Tools For Model Compression

TensorFlow Lite

EDGE IMPULSE

arm

MICROEJ

STM32 Cube.AI

NVIDIA JETSON

Model Evaluation : Performance Metrics

Model Evaluation : Performance Metrics

- Confusion Matrix
- Precision, Recall, and F1 Score
- Balanced Accuracy
- Receiver Operator Characteristics Curve (ROC)

Terminology

Often used in Pattern Classification Problems:

True positive

- The object is there and our classifier says it is there

True negative

- The object is not there and our classifier says it is not there

False negative (false misses)

- The object is there and our classifier says it is not there

False positive (false hits)

- The object is not there and our classifier says it is there

2 x 2 Confusion Matrix

	PREDICTED CLASS	
	P	N
ACTUAL CLASS P	TRUE POSITIVE (TP)	FALSE NEGATIVE(FN)
ACTUAL CLASS N	FALSE POSITIVE (FP)	TRUE NEGATIVE(TN)

ERR = (FP + FN) / (FP + FN + TP + TN) = 1 - ACC

ACC = (TP + TN)/ (FP + FN + TP + TN) = 1- ERR

Limitation of Accuracy

Consider a 2-class problem

- Number of Class 0 examples = 9990
- Number of Class 1 examples = 10

If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %

- Accuracy is misleading because model does not detect any class 1 example

True Positive Rate and False Positive Rate

$$\textbf{TPR} = \frac{\textbf{TP}}{\textbf{P}} = \frac{\textbf{TP}}{\textbf{TP+FN}} = \textbf{1- FNR}$$

The true positive rate represents the proportion of observations that are predicted to be positive when indeed they are positive.

$$\textbf{FPR} = \frac{\textbf{FP}}{\textbf{N}} = \frac{\textbf{FP}}{\textbf{FP+TN}} = \textbf{1 – TNR}$$

False positive rate represents the proportion of observations that are predicted to be positive when they’re actually negative.

False Negative Rate and True Negative Rate

$$\text{FNR} = \frac{FN}{P} = \frac{\text{FN}}{FN+TP} = 1 - \text{TPR}$$

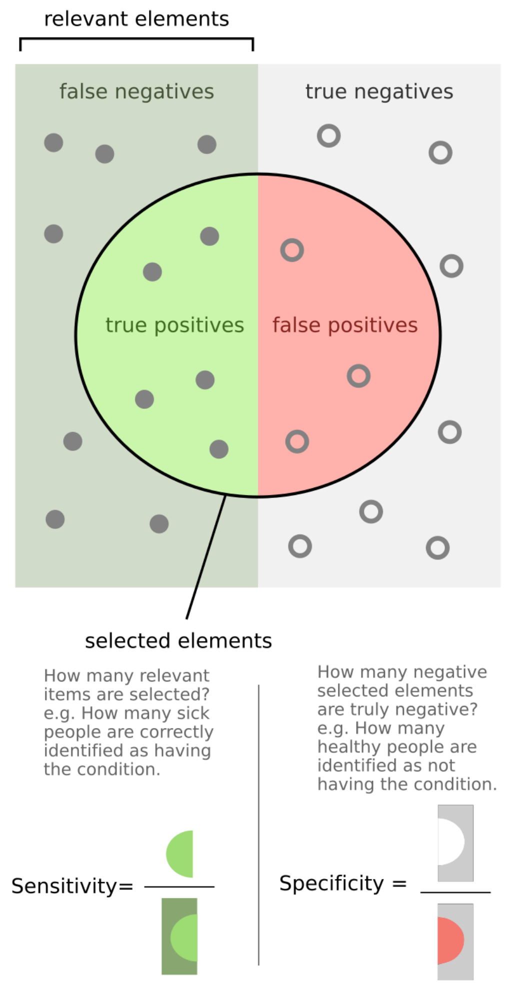
$$\text{TNR} = \frac{TN}{N} = \frac{\text{TN}}{TN+FP} = 1 - \text{FPR}$$

Sensitivity and Specificity

- **Sensitivity** (True Positive Rate) is the probability of a positive test result, conditioned on the individual truly being positive.
  - Sensitivity = TPR = $\frac{\mathbf{TP}}{\mathbf{P}} = \frac{\mathbf{TP}}{\mathbf{TP+FN}}$

- **Specificity** (True Negative Rate) is the probability of a negative test result, conditioned on the individual truly being negative.
  - Specificity = TNR = $\frac{\boldsymbol{TN}}{\boldsymbol{N}} = \frac{\mathbf{TN}}{\boldsymbol{TN+FP}}$

Sensitivity and Specificity



Sensitivity (True Positive Rate)

Specificity (True Negative Rate)

Sensitivity and Specificity

- **Sensitivity**
  - Probability of a true-positive = TP/(TP+FN)
- **Specificity**
  - Probability of a true-negative = TN/(TN+FP)

- **The probability of a correct decision = (TP+TN)/S, where S is the total number of samples**

Precision, Recall

Precision is the ratio between true positives versus all positives,

$$\text{Precision} = \frac{\boldsymbol{T}\mathbf{P}}{\mathbf{TP}+\boldsymbol{FP}}$$

Recall is the measure of how accurate the model is in identifying true positives

$$\text{Recall} = \frac{\mathbf{TP}}{\mathbf{TP}+\mathbf{FN}}$$

Precision, Recall, Accuracy

Precision = TP/(TP+FP)

Recall = TP/(TP+FN)

Accuracy = (TP+TN)/(TP+TN+FP+FN)

F1 Score

F1 score takes into account both precision and recall and is based on a balance of the two.

$$F1 = \frac{2}{\frac{1}{P}+\frac{1}{R}} = \frac{2PR}{P+R}$$

The F1 score is a useful performance metric in machine learning, especially for imbalanced classification problems, where the number of samples in each class is not equal. The F1 score combines two important metrics: precision and recall, into a single value, helping to balance their trade-off.

Balanced Accuracy

Balanced accuracy provides a more insightful measure by accounting for both your model's sensitivity (true positive rate) and specificity (true negative rate). This makes it particularly valuable in real-world scenarios where imbalanced data is common, and the minority class is usually more important.

Balanced Accuracy

- Balanced Accuracy accounts for the performance on both the positive and negative classes, making it particularly useful in imbalanced datasets.
- It is the average of Sensitivity(Recall) and Specificity (True Negative Rate).

$$\textbf{Balanced Accuracy} = \frac{\boldsymbol{Sensitivity+Specificity}}{\mathbf{2}}$$

- Standard Accuracy may be misleading in imbalanced datasets because a model could achieve high accuracy simply by predicting the majority class, while balanced accuracy ensures both classes are considered equally.

Parameters vs. Performance

- Once we have designed our classifier, we invariably have some parameters we'd like to adjust. e.g.
  - Prior probability, Threshold

- The optimal classifier is one with sensitivity (Probability of True Positive) as close to 100% as possible, and at the same time with specificity (Probability of True Negative) as close to 100% as possible

ROC – Receiver Operating Characteristic

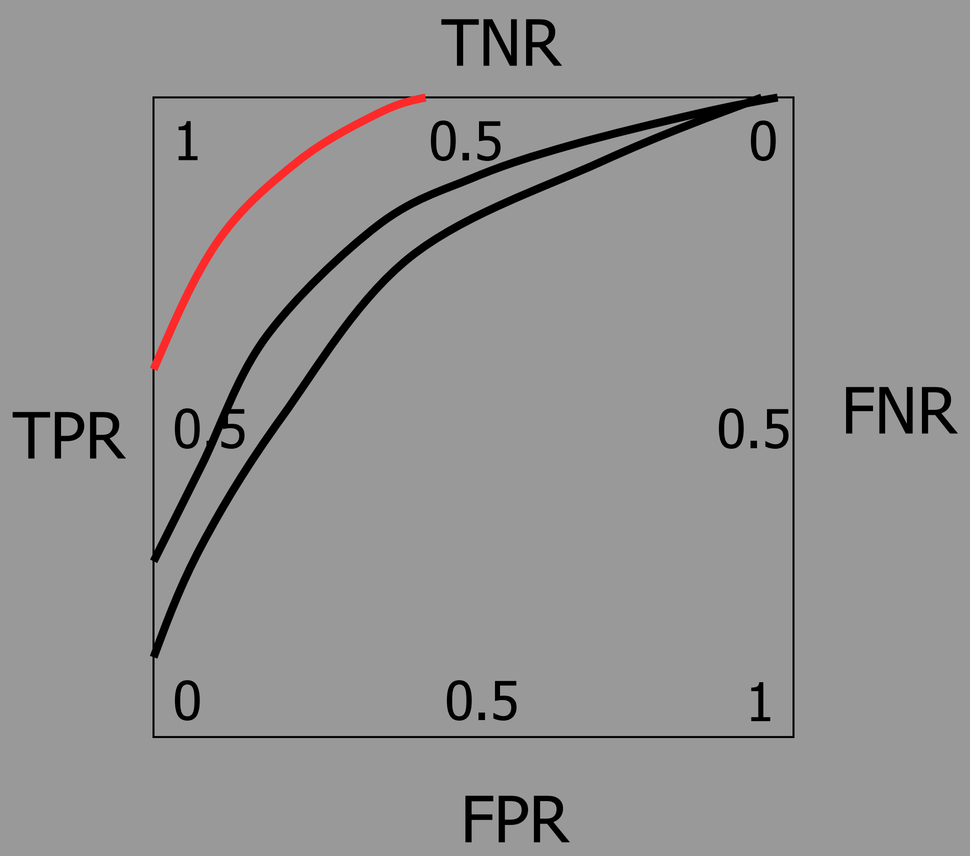
- **Developed in 1950s for signal detection theory to analyze noisy signals**
  - Characterize the trade-off between positive hits and false alarms
- **ROC curve plots TP (on the y-axis) against FP (on the x-axis)**
- **Performance of each classifier represented as a point on the ROC curve**
  - changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point

ROC – Receiver Operating Characteristic

- A Receiver Operating Characteristic Curve (ROC) is a standard technique for summarizing classifier performance over a range of trade-offs between true positive (TP) and false positive (FP) error rates (Sweets, 1988).

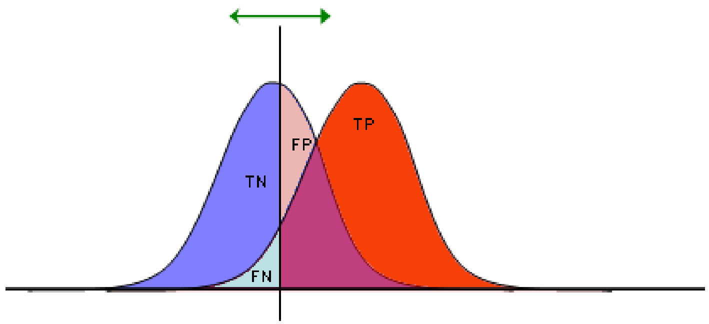
- ROC curve is a plot of sensitivity (the ability of the model to predict an event correctly) versus 1-specificity for the possible cut-off classification probability values .

ROC – Receiver Operating Characteristic

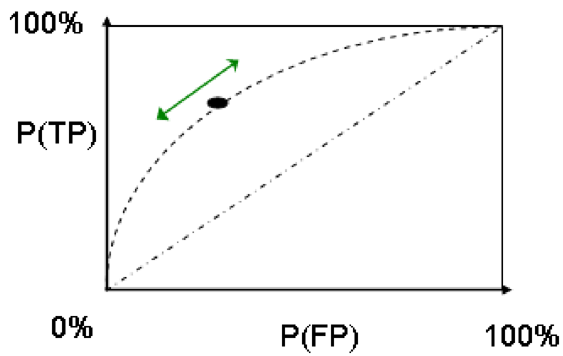


- Each curve represents the performance of a particular classifier as some parameter is varied over its range.
- Of the three curves, the one with the sharpest bend, which passes closest to the upper left corner, is the best
- Calculate the area under the curve, AUC and the one with highest value is the best, or calculate the area above the curve, the one with the smallest area is the best.
- TPR: TP out of the total actual positives (Sensitivity or Recall)
- FPR: FP out of the total actual negatives (1-Specificity)

ROC – Receiver Operating Characteristic



TP	FP
FN	TN
1	1



http://en.wikipedia.org/wiki/Receiver_operating_characteristic

Acknowledgement

The material in the lecture is prepared with the help of material from Coursera course “Introduction to Embedded Machine Learning” and HarvardX.

https://tinyml.seas.harvard.edu/courses/