CS249r: Overview and Introduction to TinyML

Vijay Janapa Reddi, Ph. D. | Associate Professor |
John A. Paulson School of Engineering and Applied Sciences | Harvard University |
Web: http://scholar.harvard.edu/vijay-janapa-reddi

Goals for today

- Introductions
- Course logistics
- Introduction to TinyML

Introductions

About Me

- Graduated from Harvard in 2010
- Professor at UT Austin
- Returned to Harvard
- Research
    - Applied machine learning architect
    - Computer architecture and runtime systems
    - Embedded hardware and software

Teaching Staff

Matthew Stewart

Postdoctoral Researcher,

Harvard University

Ikechukwu Uchendu

Computer Science, PhD

Student, Harvard University

Jason Jabbour

Computer Science, PhD

Student, Harvard University

Jessica Quaye

Computer Science, PhD

Student, Harvard University

Course Logistics

What you will learn

Through this course, you would have been exposed to the following:

- Brief introduction to ML and IoT
- Industry talks about real-world deployments
- Discussion of bleeding edge academic research
- Practical experience through hands-on project assignments

At the end of the course, you would have achieved the following:

- Gained familiarity with cutting edge literature in the field of tinyML
- Learnt to train and deploy models on microcontrollers with TF-micro
- Conceived and developed a (novel) TinyML application running on a MCU

Course Topics

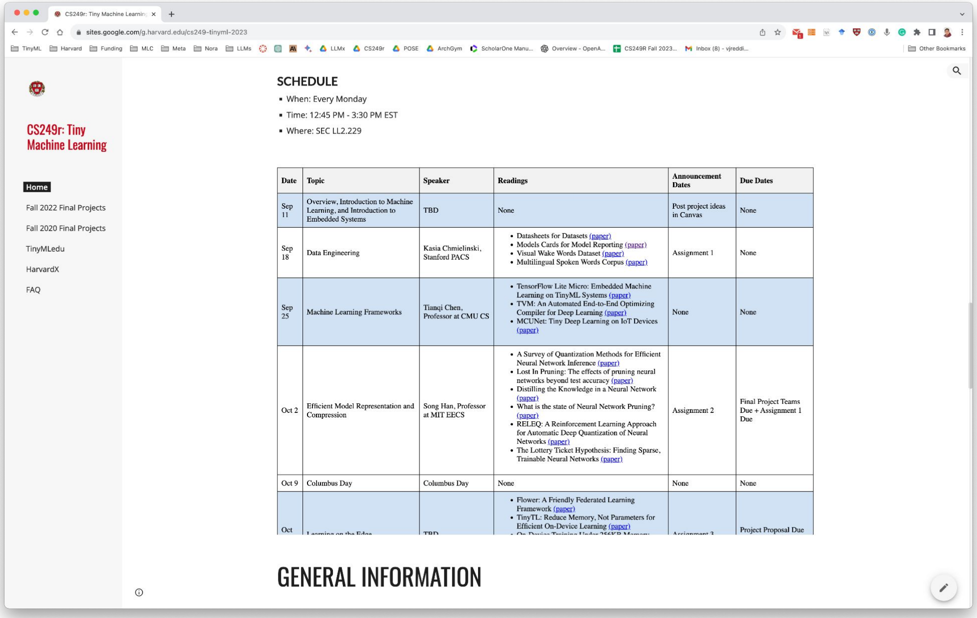
1. **Overview and Introduction to Embedded Machine Learning**
2. Data Engineering
3. Embedded Machine Learning Frameworks
4. Efficient Model Representation and Compression
5. Performance Metrics and Benchmarking of ML Systems
6. Learning on the Edge
7. Hardware Acceleration for Edge ML: GPUs, TPUs and FPGAs
8. Embedded MLOps
9. Secure and Privacy-Preserving On-Device ML
10. Responsible AI
11. Sustainability at the Edge
12. Generative AI at the Edge

Grading

Class Participation	= 10%
Paper Reviews	= 10%
Paper Presentation	= 10%
Programming Assignments	= 25%
Final Project	= 45%

Paper Readings

- Check to see the **bold** list of papers, these are the ones to read and come prepared for discussion
- Discuss in groups
- One lead per paper
- Report back to the group



CS249r: Tiny Machine Learning

Home

Fall 2022 Final Projects

Fall 2020 Final Projects

TinyMLedu

HarvardX

FAQ

SCHEDULE

- When: Every Monday
- Time: 12:45 PM - 3:30 PM EST
- Where: SEC LL2.229

Date	Topic	Speaker	Readings	Announcement Dates	Due Dates
Sep 11	Overview, Introduction to Machine Learning, and Introduction to Embedded Systems	TBD	None	Post project ideas in Canvas	None
Sep 18	Data Engineering	Kasia Chmielinski, Stanford PACS	• Datasheets for Datasets (paper) • Models Cards for Model Reporting (paper) • Visual Wake Words Dataset (paper) • Multilingual Spoken Words Corpus (paper)	Assignment 1	None
Sep 25	Machine Learning Frameworks	Tianqi Chen, Professor at CMU CS	• TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems (paper) • TVM: An Automated End-to-End Optimizing Compiler for Deep Learning (paper) • MCUNet: Tiny Deep Learning on IoT Devices (paper)	None	None
Oct 2	Efficient Model Representation and Compression	Song Han, Professor at MIT EECS	• A Survey of Quantization Methods for Efficient Neural Network Inference (paper) • Lost In Pruning: The effects of pruning neural networks beyond test accuracy (paper) • Distilling the Knowledge in a Neural Network (paper) • What is the state of Neural Network Pruning? (paper) • RELEQ: A Reinforcement Learning Approach for Automatic Deep Quantization of Neural Networks (paper) • The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks (paper)	Assignment 2	Final Project Teams Due + Assignment 1 Due
Oct 9	Columbus Day	Columbus Day	None	None	None
Oct	Learning on the Edge	TBD	• Flower: A Friendly Federated Learning Framework (paper) • TinyTL: Reduce Memory, Not Parameters for Efficient On-Device Learning (paper) • On-Device Training Under 256KB Memory	Assignment 3	Project Proposal Due

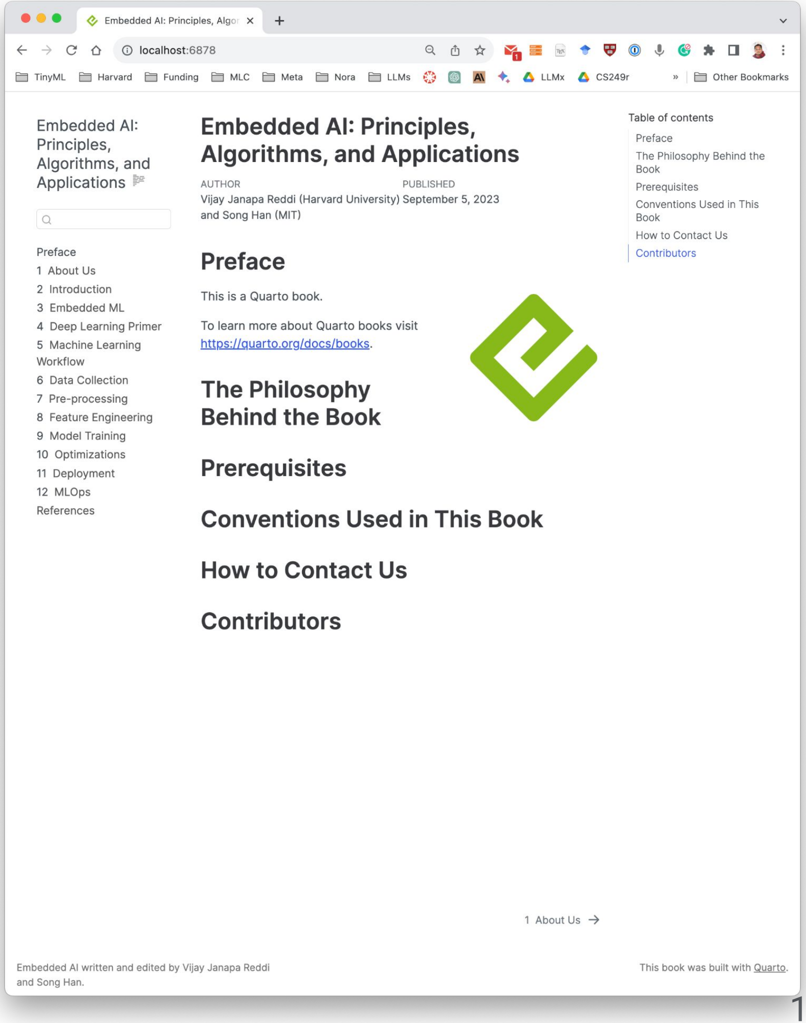
GENERAL INFORMATION

Paper Reviews & Presentations

Goal: Share the knowledge

- Listen to the lectures
- Read the papers
- Review the material
- Scribe into the open source book

Instructions will be announced soon!



Embedded AI: Principles, Algorithms, and Applications

Embedded AI: Principles, Algorithms, and Applications

AUTHOR
Vijay Janapa Reddi (Harvard University) and Song Han (MIT)

PUBLISHED
September 5, 2023

Preface

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

The Philosophy Behind the Book

Prerequisites

Conventions Used in This Book

How to Contact Us

Contributors

Table of contents

1 About Us →

Embedded AI written and edited by Vijay Janapa Reddi and Song Han.

This book was built with Quarto.

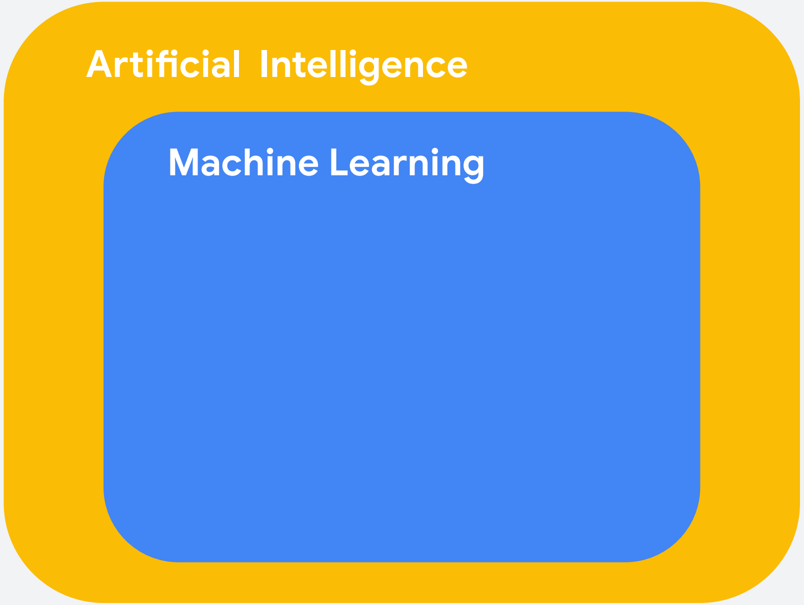
Class Schedule

		Start	~End
Lecture		12:45:00 PM	1:30:00 PM
Break		1:40:00 PM	1:45:00 PM
Paper Discussions	Paper 1	1:45:00 PM	2:05:00 PM
	Paper 2	2:05:00 PM	2:25:00 PM
Break		2:25:00 PM	2:30:00 PM
Guest Lecture		2:30:00 PM	3:30:00 PM

Introduction to TinyML

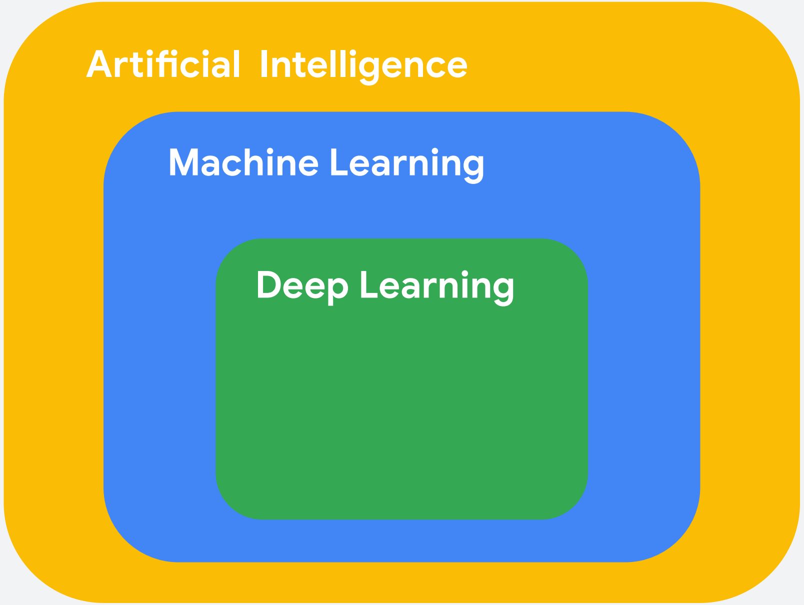
What is Machine Learning?

1. **Machine Learning** is a subfield of **Artificial Intelligence** focused on developing algorithms that learn to **solve problems by analyzing data for patterns**



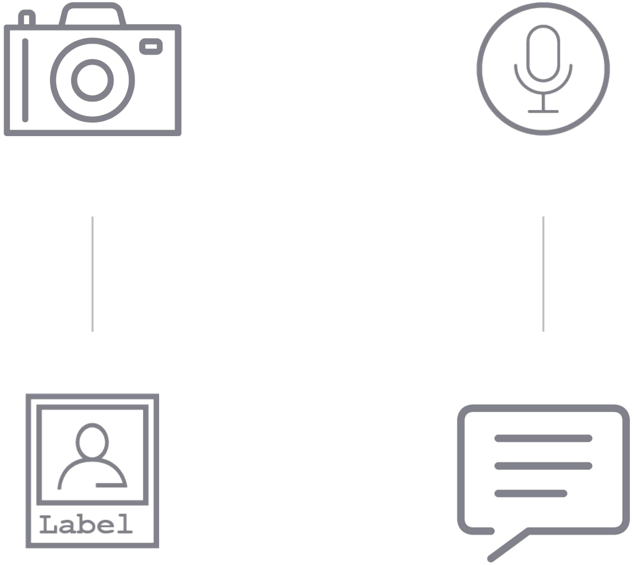
What is (Deep) Machine Learning?

1. Machine Learning is a subfield of Artificial Intelligence focused on developing algorithms that learn to solve problems by analyzing data for patterns
2. **Deep Learning** is a type of Machine Learning that leverages **Neural Networks** and **Big Data**



Applications of Machine Learning

Applications of Machine Learning



Applications of Machine Learning

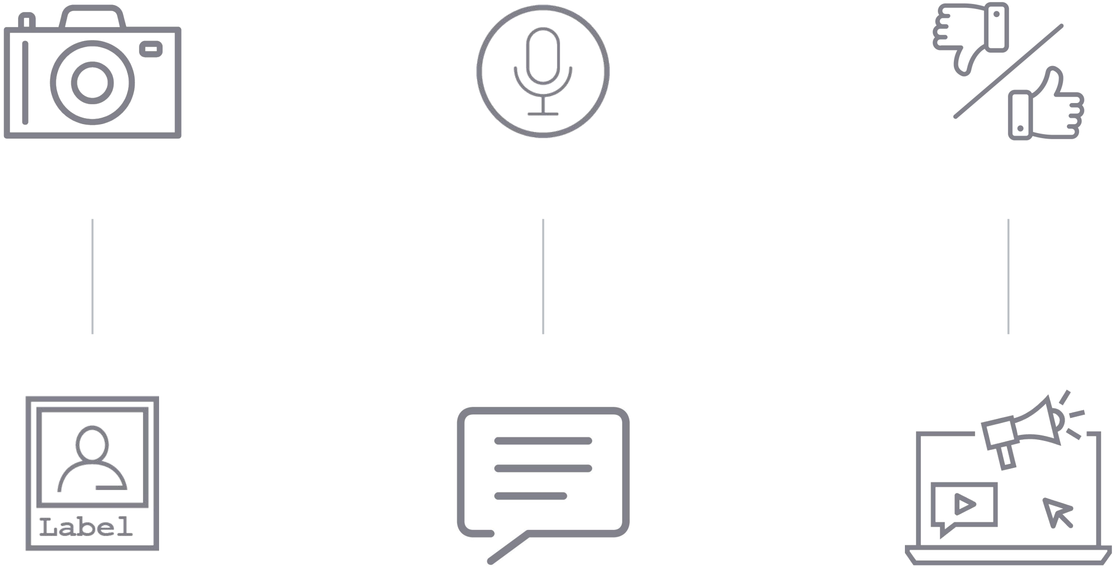
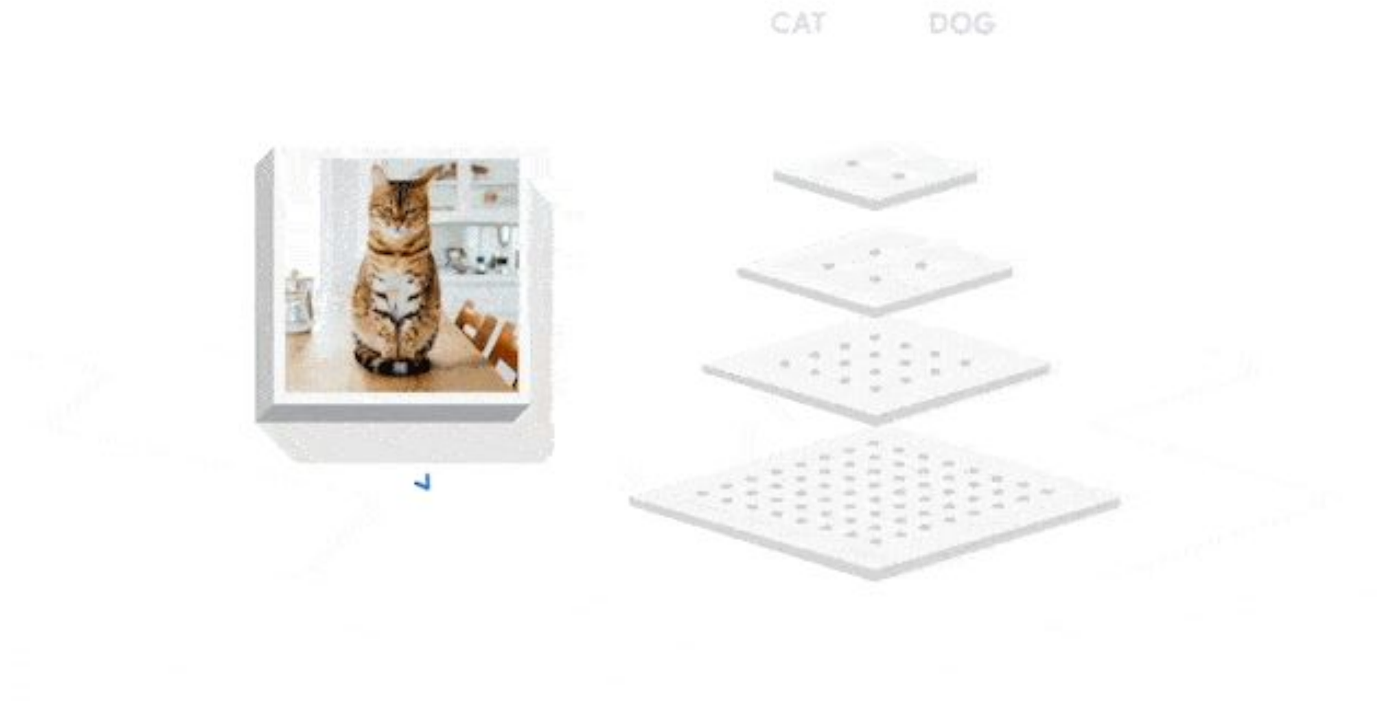
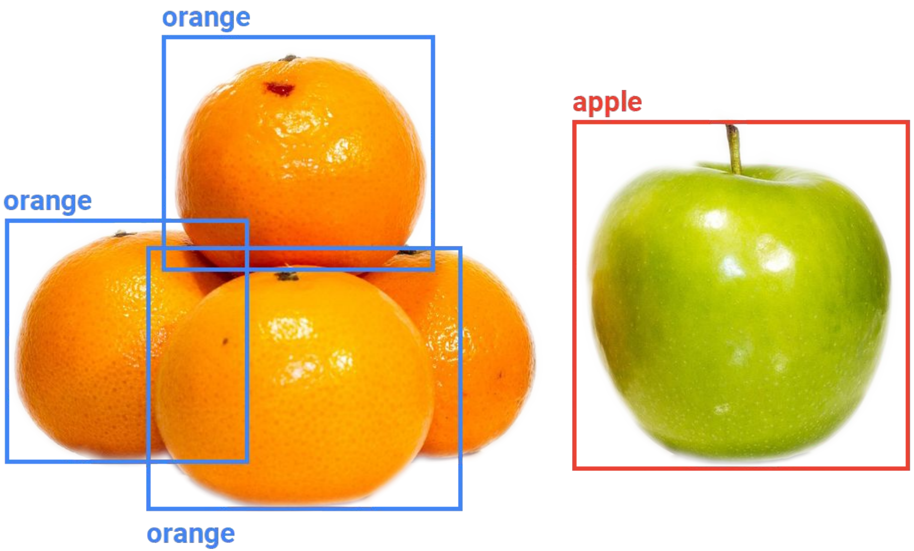


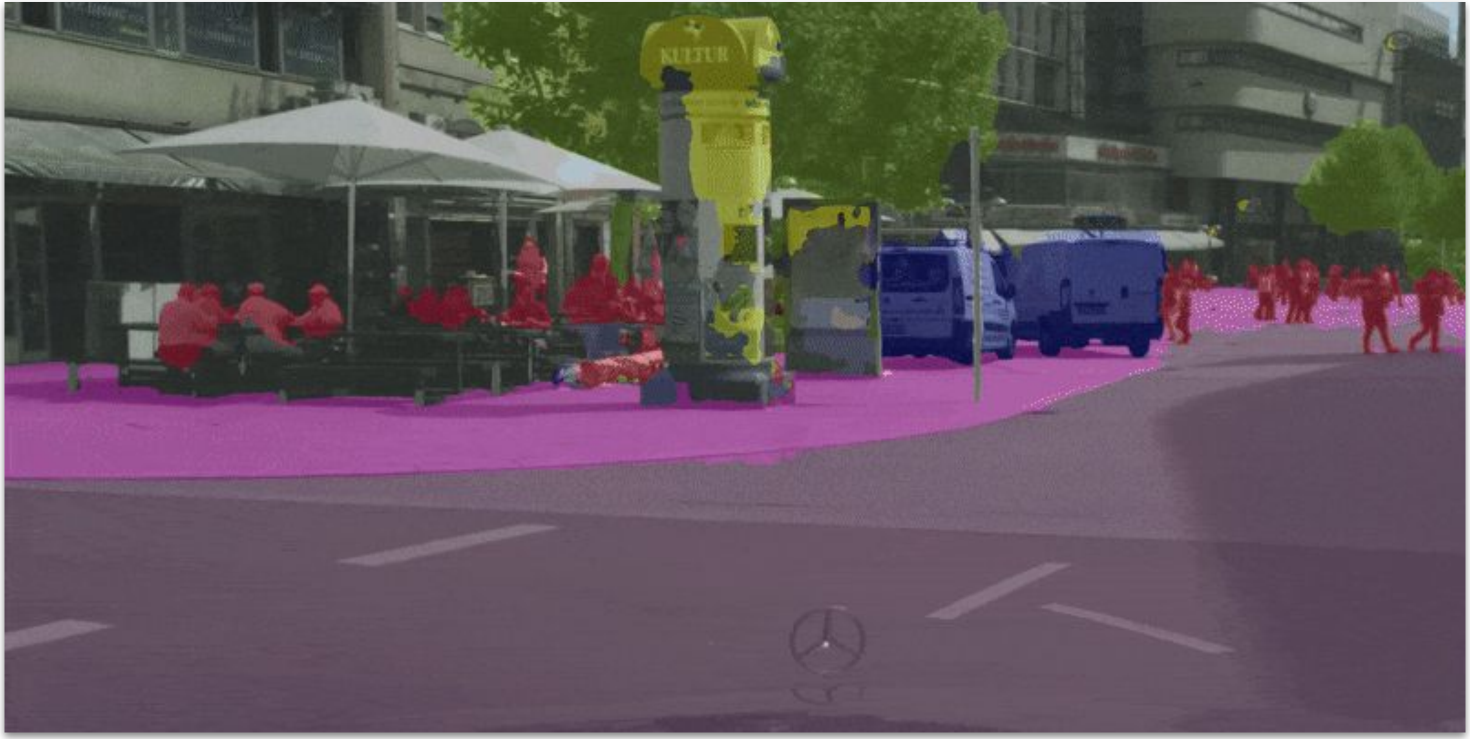
Image Classification



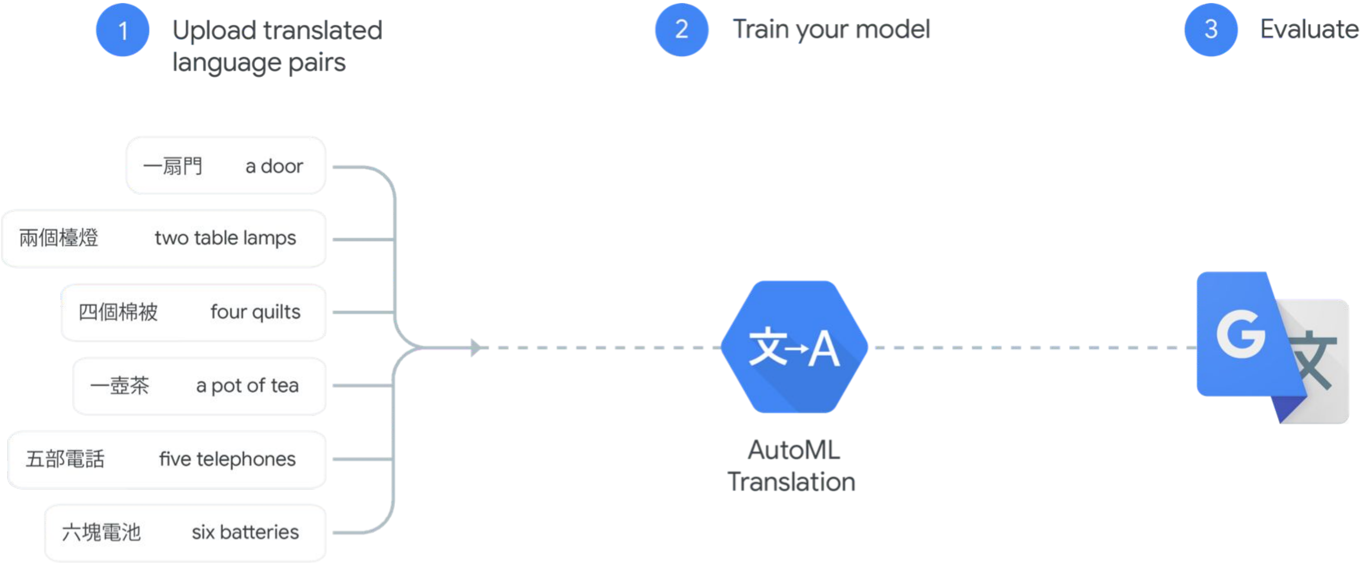
Object Detection



Segmentation



Machine Translation



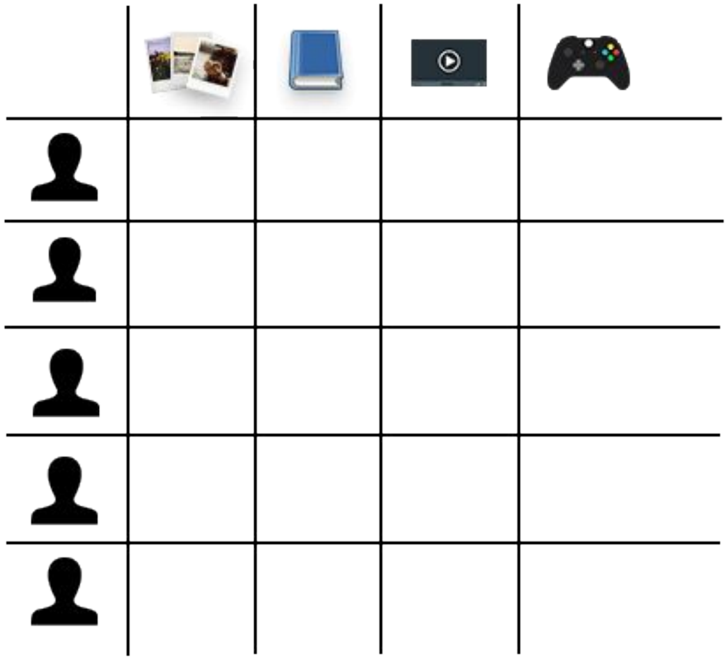
1 Upload translated language pairs

2 Train your model

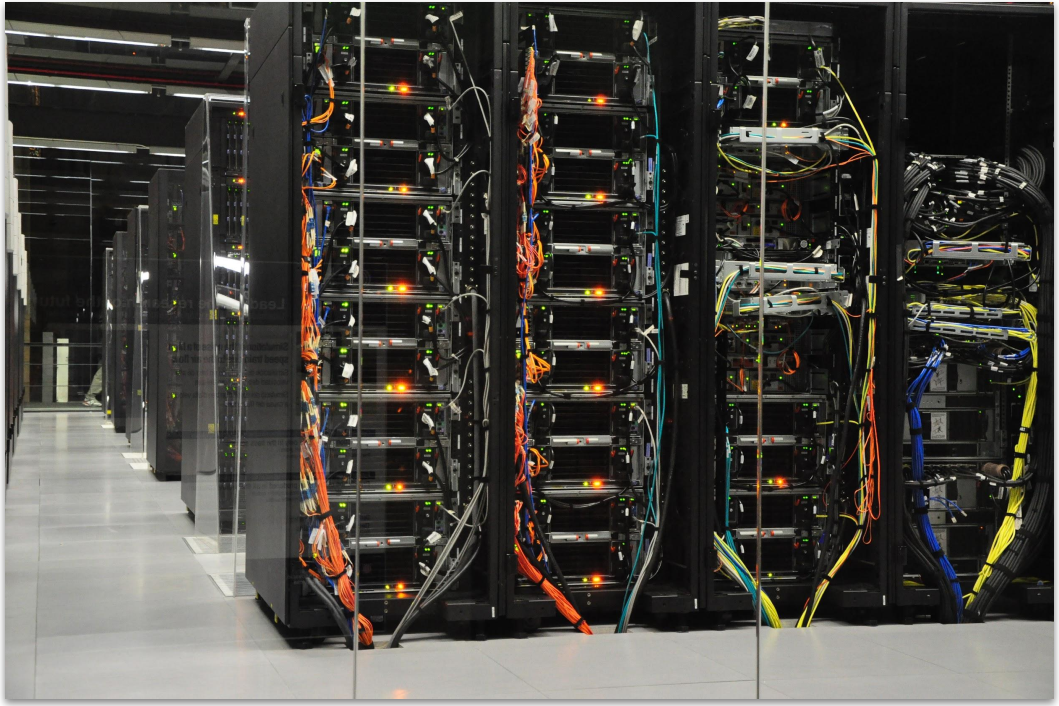
3 Evaluate

一扇門	a door
兩個檯燈	two table lamps
四個棉被	four quilts
一壺茶	a pot of tea
五部電話	five telephones
六塊電池	six batteries

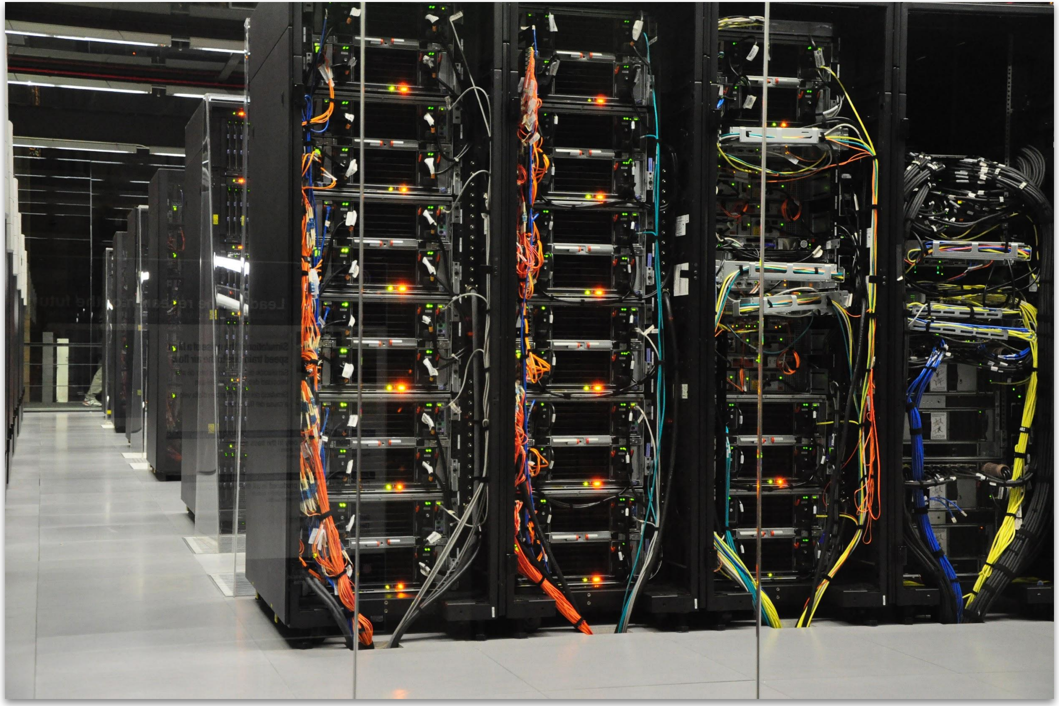
Recommendations



Datacenter



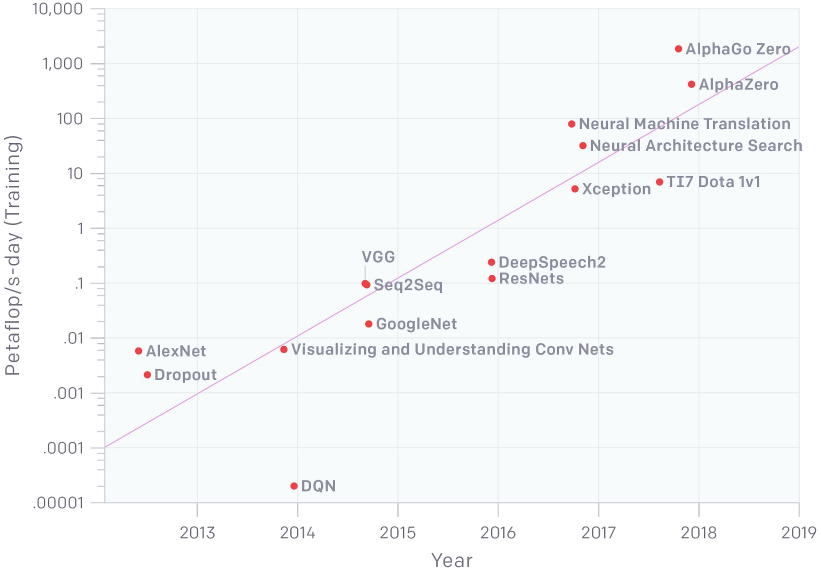
Datacenter



AI Compute: 300,000x Increase in Demand

"… **since 2012 the amount of compute used in the largest AI training runs has been increasing exponentially with a 3.5 month-doubling time** (by comparison, Moore's Law had an 18-month doubling period). Since 2012, this metric has **grown by more than 300,000x (an 18-month [Moore's Law] doubling period would yield only a 12x increase)**. Improvements in compute have been a key component of AI progress, so as long as this trend continues, it's worth preparing for the implications of systems far outside today's capabilities."

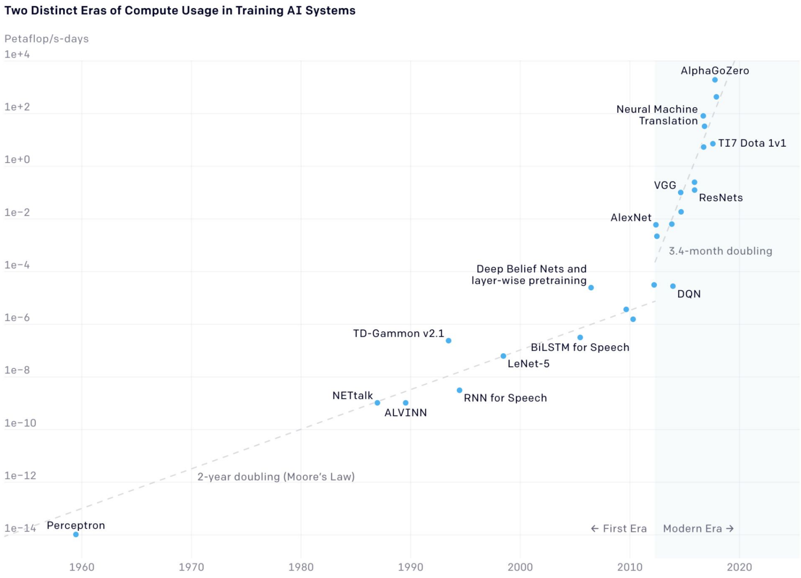
AlexNet to AlphaGo Zero: A 300,000x Increase in Compute



Source: https://blog.openai.com/ai-and-compute/

Two Eras of Computing

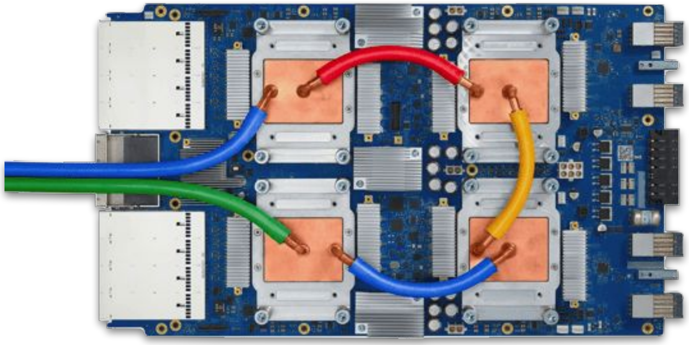
"… **since 2012 the amount of compute used in the largest AI training runs has been increasing exponentially with a 3.5 month-doubling time** (by comparison, Moore's Law had an 18-month doubling period). Since 2012, this metric has **grown by more than 300,000x (an 18-month [Moore's Law] doubling period would yield only a 12x increase)**. Improvements in compute have been a key component of AI progress, so as long as this trend continues, it's worth preparing for the implications of systems far outside today's capabilities."

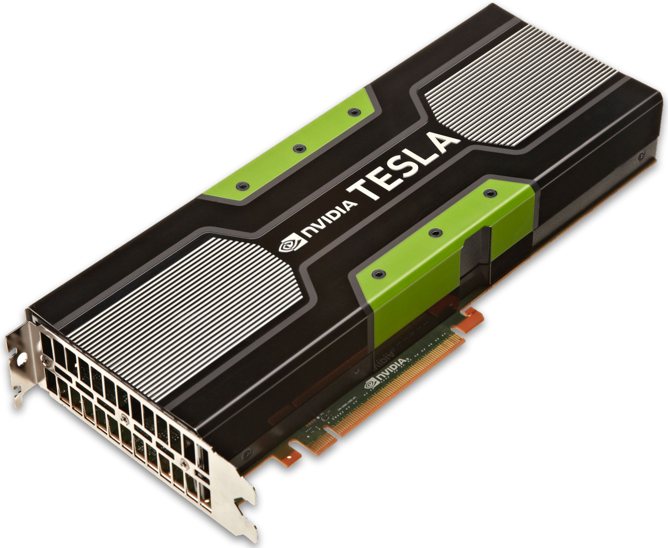


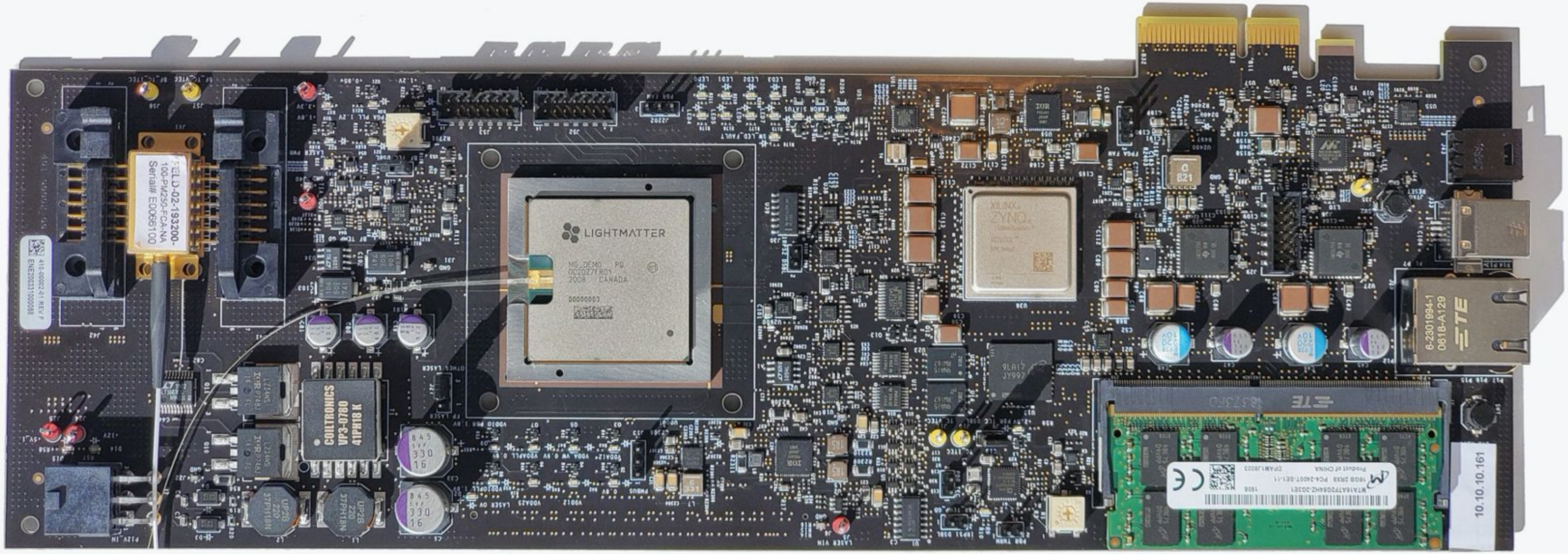
Two Distinct Eras of Compute Usage in Training AI Systems

Source: https://blog.openai.com/ai-and-compute/

TPUs/GPUs



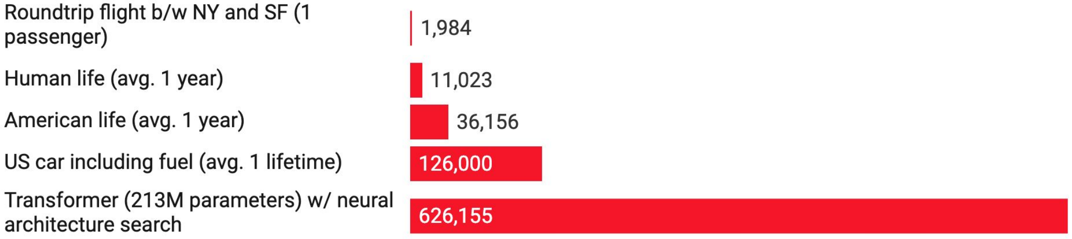


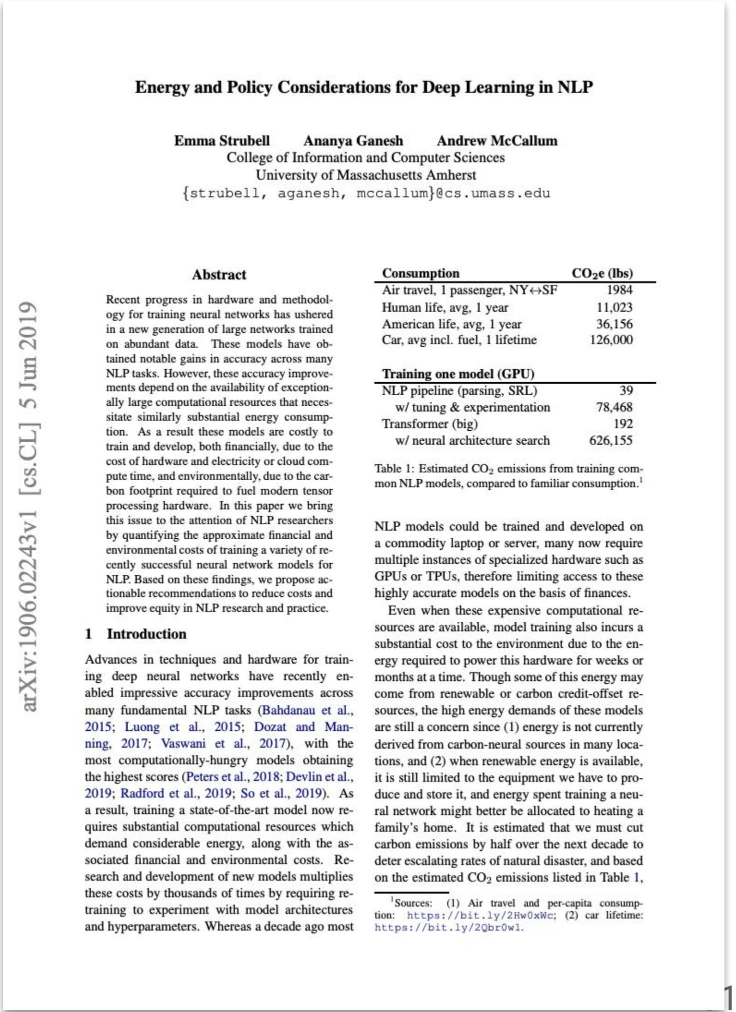

LIGHTMATTER
XILINX ZYNQ
COILTRONICS VP3-0780
TE
6-2301994-1
0818-A129
10.10.10.161

But… Bigger Is Not Always Better.

Common carbon footprint benchmarks

in lbs of CO2 equivalent





Energy and Policy Considerations for Deep Learning in NLP

Emma Strubell **Ananya Ganesh** **Andrew McCallum**
College of Information and Computer Sciences
University of Massachusetts Amherst
{strubell, aganesh, mccallum}@cs.umass.edu

arXiv:1906.02243v1 [cs.CL] 5 Jun 2019

Abstract

Recent progress in hardware and methodology for training neural networks has ushered in a new generation of large networks trained on abundant data. These models have obtained notable gains in accuracy across many NLP tasks. However, these accuracy improvements depend on the availability of exceptionally large computational resources that necessitate similarly substantial energy consumption. As a result these models are costly to train and develop, both financially, due to the cost of hardware and electricity or cloud compute time, and environmentally, due to the carbon footprint required to fuel modern tensor processing hardware. In this paper we bring this issue to the attention of NLP researchers by quantifying the approximate financial and environmental costs of training a variety of recently successful neural network models for NLP. Based on these findings, we propose actionable recommendations to reduce costs and improve equity in NLP research and practice.

Consumption	**CO_2e (lbs)**
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000
Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture search	626,155

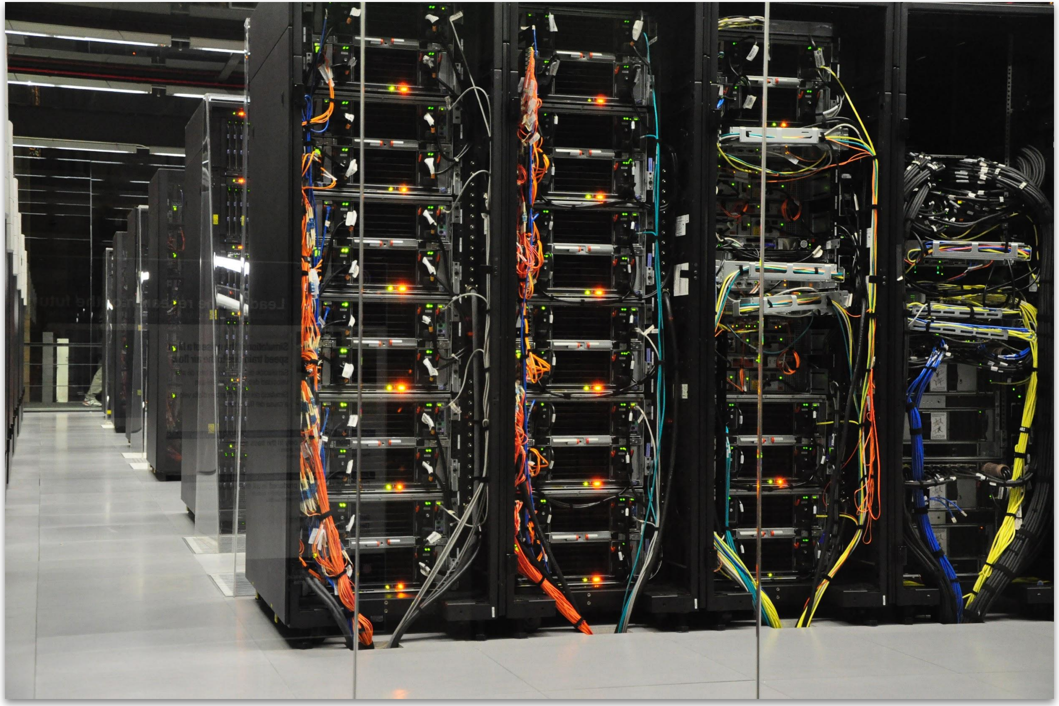
Table 1: Estimated CO_2 emissions from training common NLP models, compared to familiar consumption.[1]

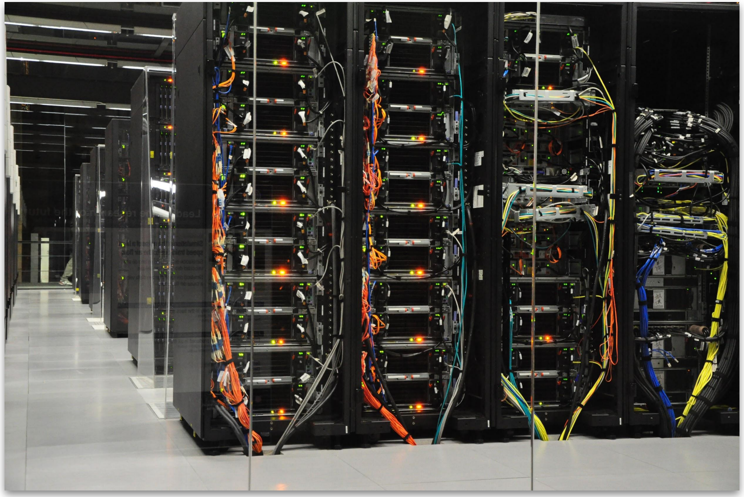
1 Introduction

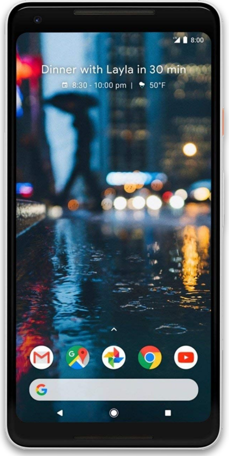
Advances in techniques and hardware for training deep neural networks have recently enabled impressive accuracy improvements across many fundamental NLP tasks (Bahdanau et al., 2015; Luong et al., 2015; Dozat and Manning, 2017; Vaswani et al., 2017), with the most computationally-hungry models obtaining the highest scores (Peters et al., 2018; Devlin et al., 2019; Radford et al., 2019; So et al., 2019). As a result, training a state-of-the-art model now requires substantial computational resources which demand considerable energy, along with the associated financial and environmental costs. Research and development of new models multiplies these costs by thousands of times by requiring retraining to experiment with model architectures and hyperparameters. Whereas a decade ago most NLP models could be trained and developed on a commodity laptop or server, many now require multiple instances of specialized hardware such as GPUs or TPUs, therefore limiting access to these highly accurate models on the basis of finances.

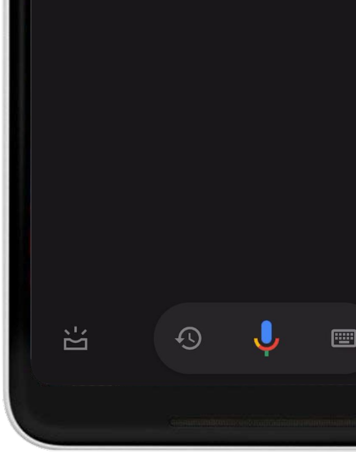
Even when these expensive computational resources are available, model training also incurs a substantial cost to the environment due to the energy required to power this hardware for weeks or months at a time. Though some of this energy may come from renewable or carbon credit-offset resources, the high energy demands of these models are still a concern since (1) energy is not currently derived from carbon-neural sources in many locations, and (2) when renewable energy is available, it is still limited to the equipment we have to produce and store it, and energy spent training a neural network might better be allocated to heating a family's home. It is estimated that we must cut carbon emissions by half over the next decade to deter escalating rates of natural disaster, and based on the estimated CO_2 emissions listed in Table 1,

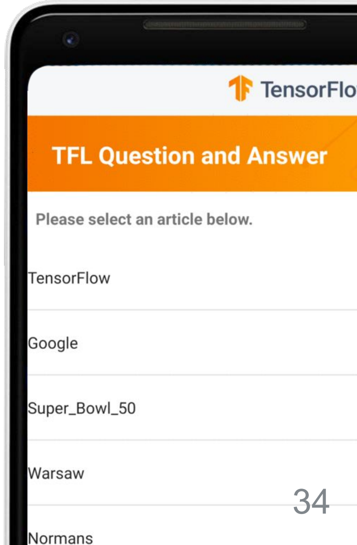
[1] Sources: (1) Air travel and per-capita consumption: https://bit.ly/2Hw0xWc; (2) car lifetime: https://bit.ly/2Qbr0w1.






8:00
Dinner with Layla in 30 min
8:30 - 10:00 pm | 50°F





TensorFlo

TFL Question and Answer

Please select an article below.

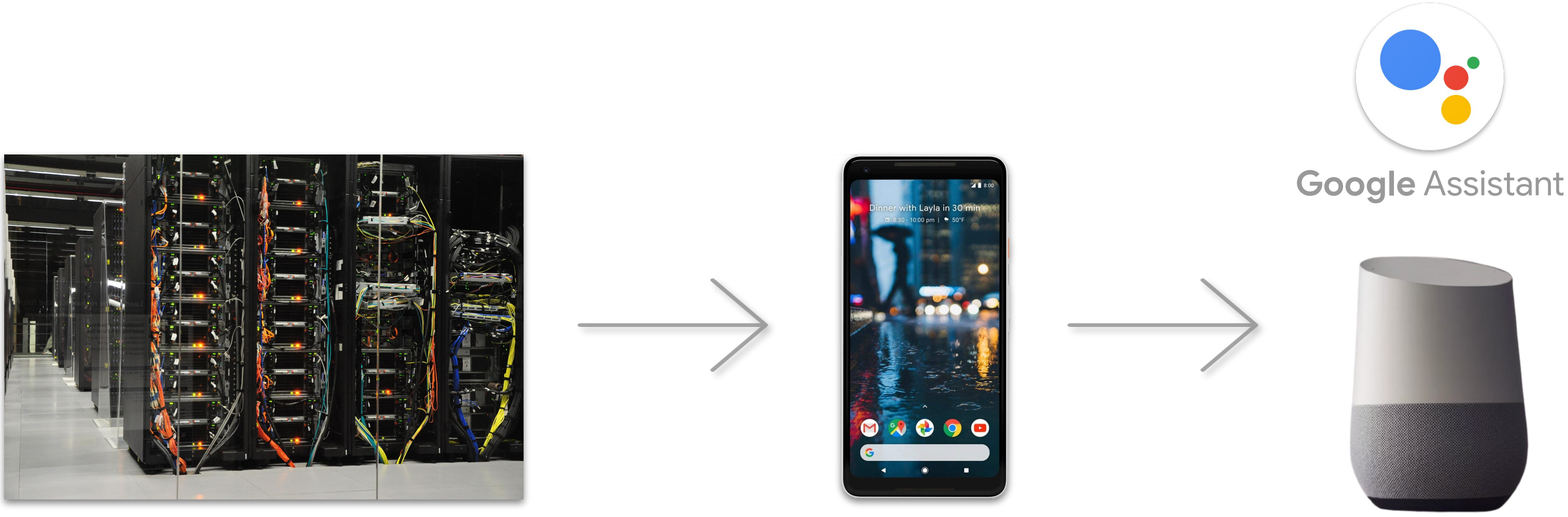
TensorFlow

Google

Super_Bowl_50

Warsaw

Normans



Google Assistant

Kicking

Penalty kicking

Passing

Dribbling

...

No Good Data Left Behind

5 Quintillion

bytes of data produced every day by IoT

<1%

of unstructured data is analyzed 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

What is TinyML?

TinyML

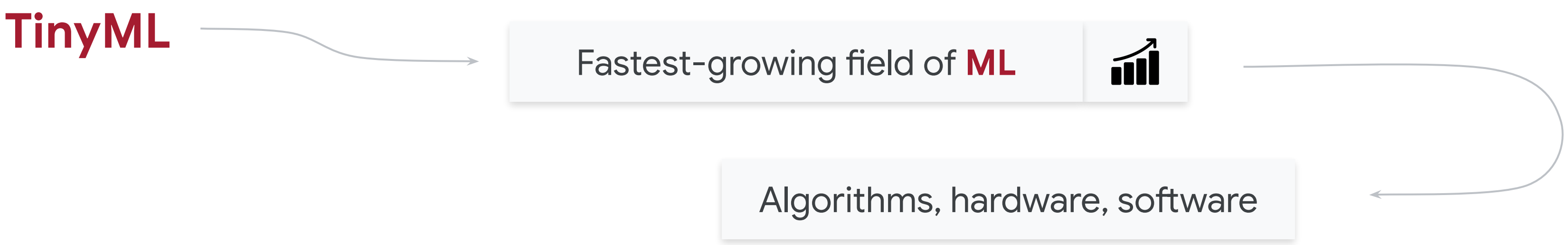
What is Tiny Machine Learning (**TinyML**)?

TinyML

What is Tiny Machine Learning (**TinyML**)?

TinyML → Fastest-growing field of **ML**

What is Tiny Machine Learning (**TinyML**)?

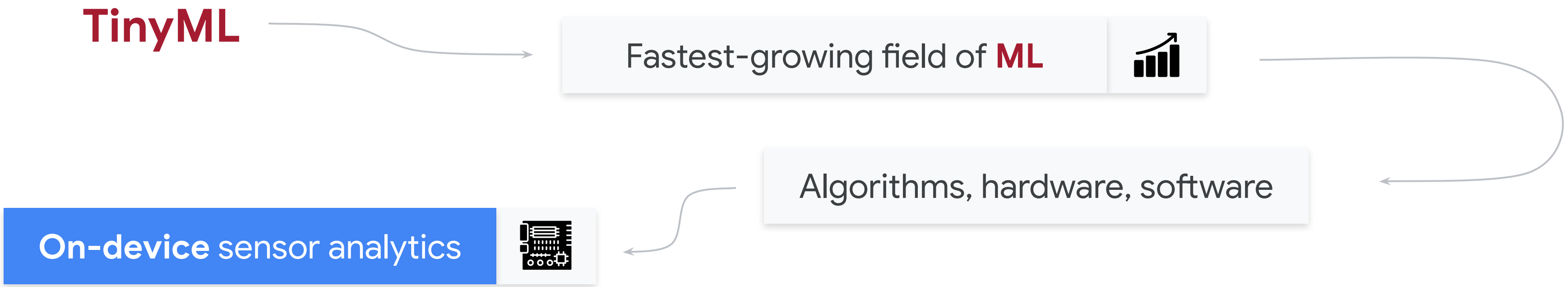


TinyML

Fastest-growing field of **ML**

Algorithms, hardware, software

What is Tiny Machine Learning (**TinyML**)?



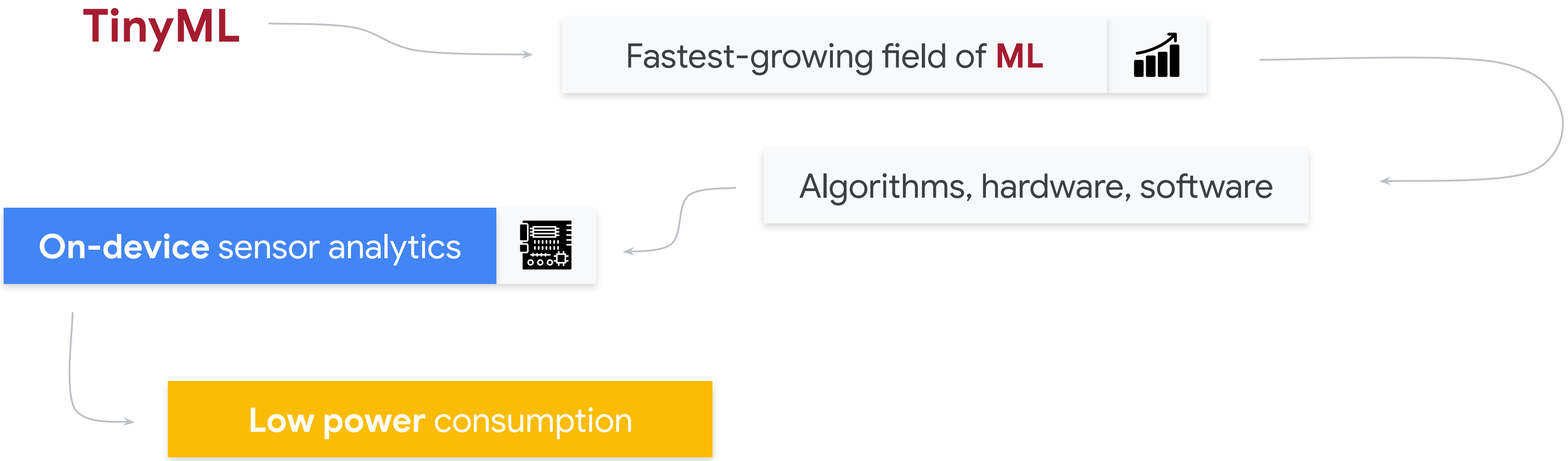
TinyML

Fastest-growing field of **ML**

Algorithms, hardware, software

On-device sensor analytics

What is Tiny Machine Learning (**TinyML**)?



TinyML

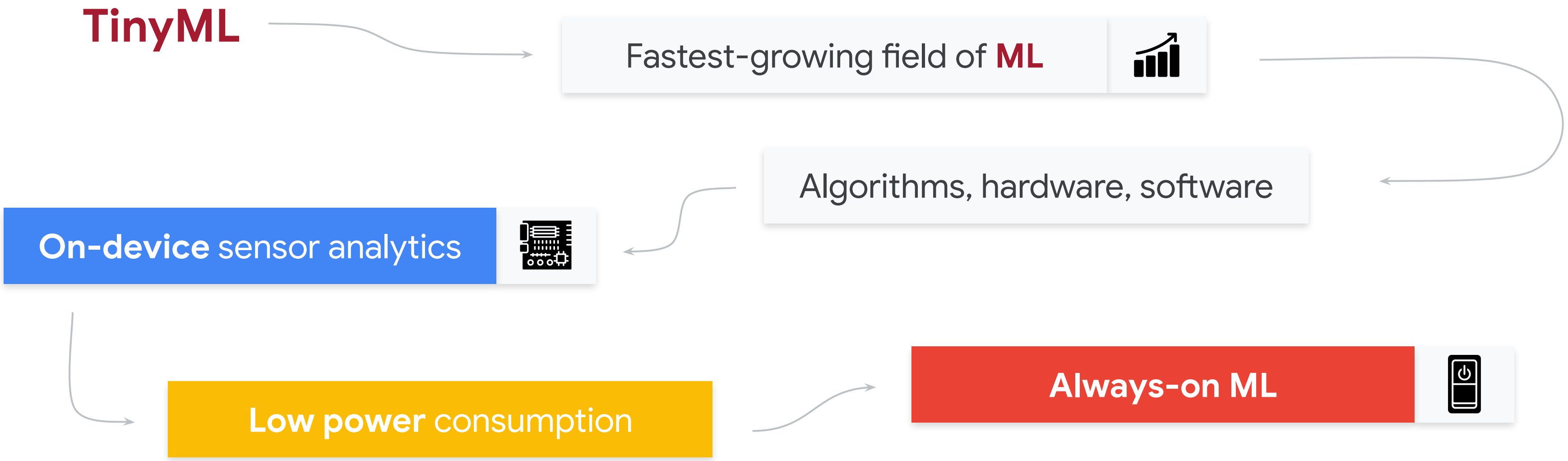
Fastest-growing field of **ML**

Algorithms, hardware, software

On-device sensor analytics

Low power consumption

What is Tiny Machine Learning (**TinyML**)?



TinyML

Fastest-growing field of **ML**

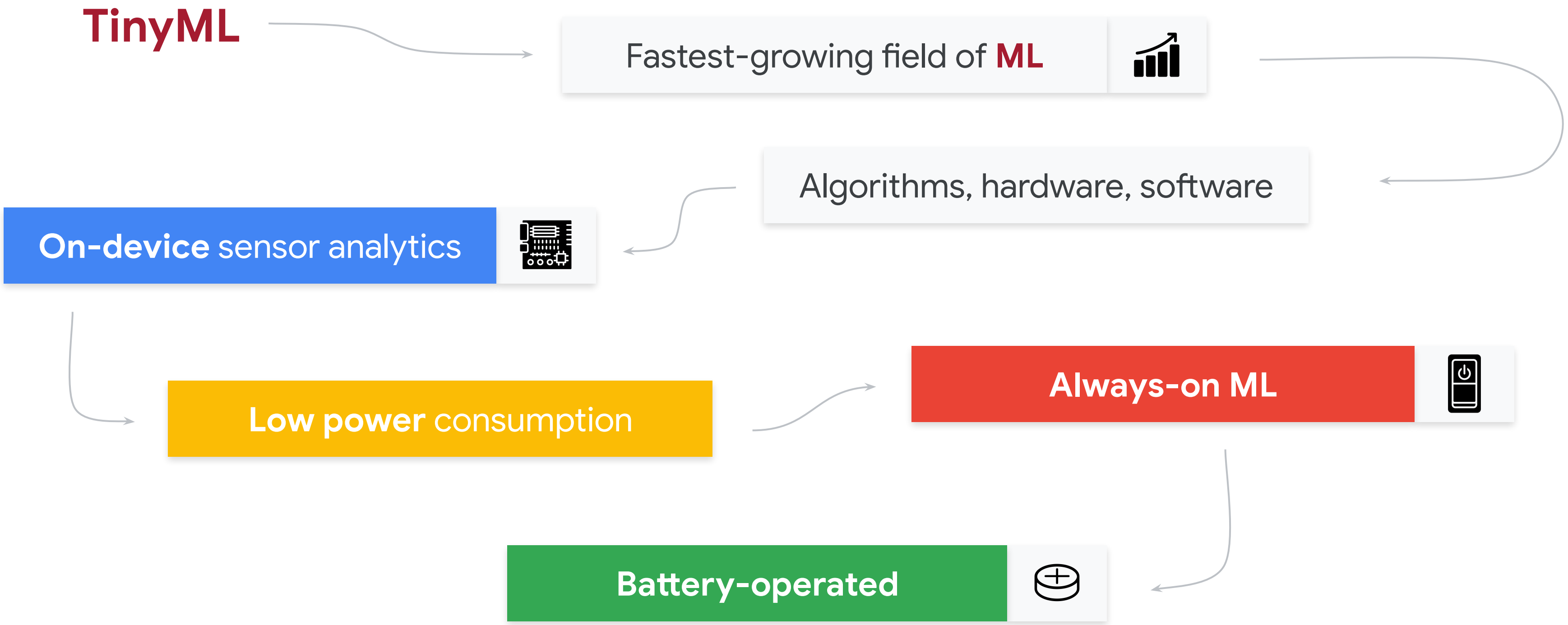
Algorithms, hardware, software

On-device sensor analytics

Low power consumption

Always-on ML

What is Tiny Machine Learning (**TinyML**)?



TinyML

Fastest-growing field of **ML**

Algorithms, hardware, software

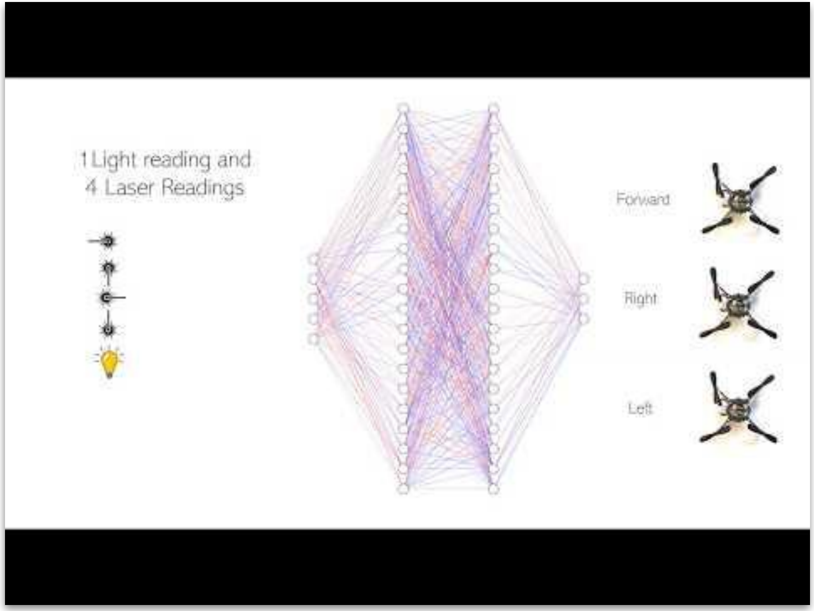
On-device sensor analytics

Low power consumption

Always-on ML

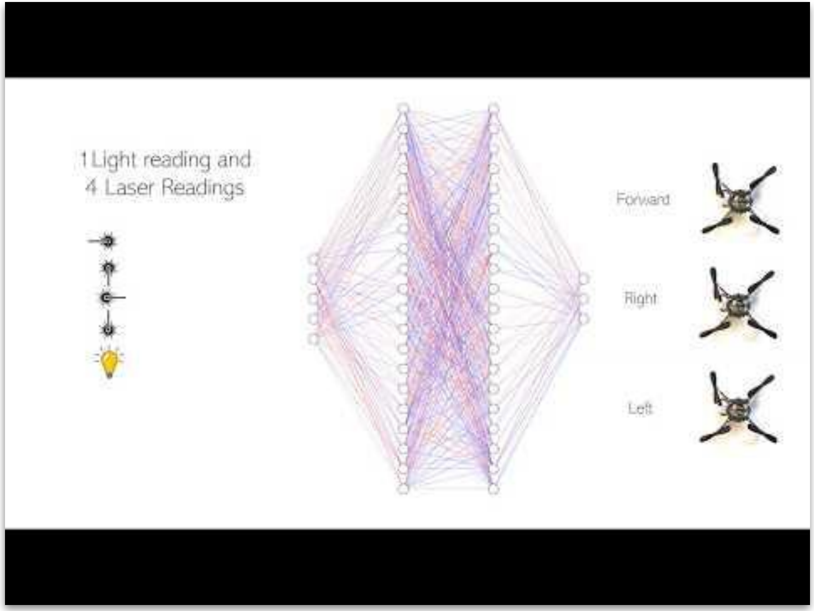
Battery-operated

Tiny Robot Learning



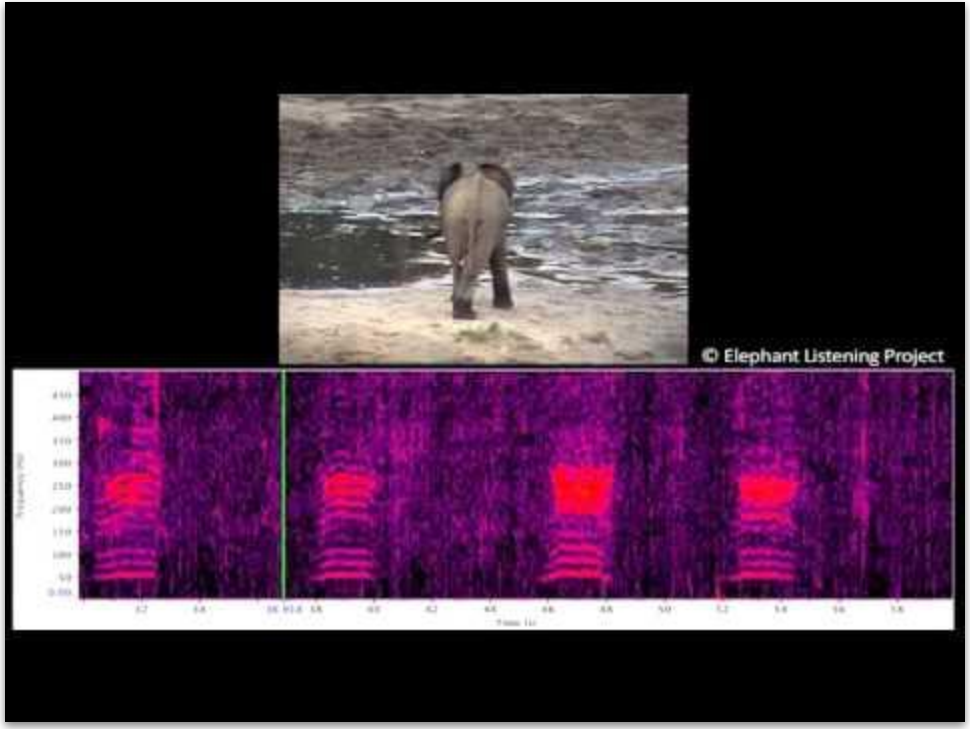
Usually, it is very challenging to get such
tiny drones to fly autonomously.

Tiny Robot Learning



Usually, it is very challenging to get such
tiny drones to fly autonomously.

Wildlife Conservation



© Elephant Listening Project

ElephantEdge

Risk Monitoring

"Know when an elephant is moving into a high-risk area and send real-time notifications to park rangers."

Conflict Monitoring

"Sense and alert when an elephant is heading into an area where farmers live."

Activity Monitoring

"Classify the general behavior of the elephant, such as when it is drinking, eating, sleeping, etc."

Communication Monitoring

"Listen for vocal communications between elephants via the onboard microphone."

Rich Array 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

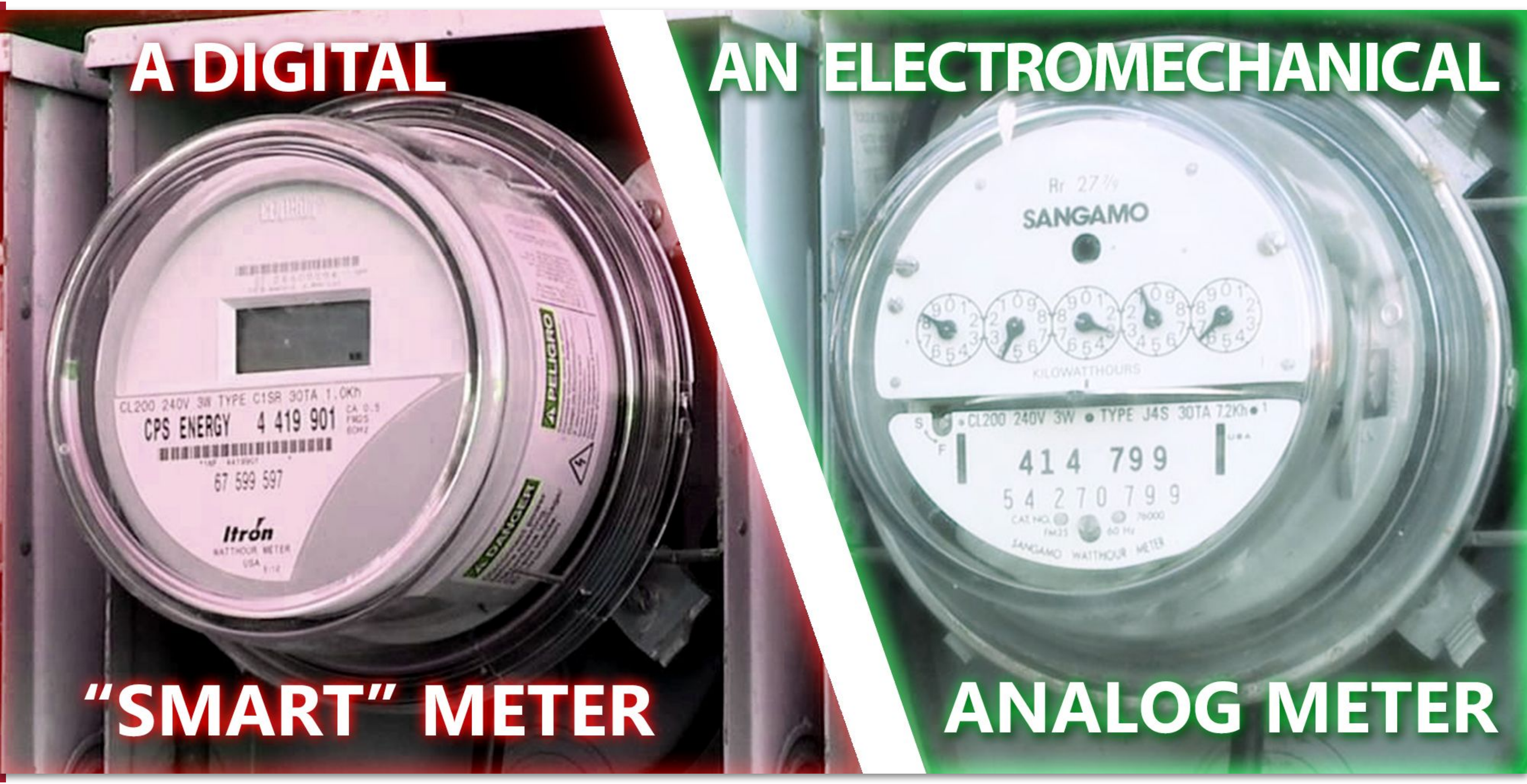
Image Sensors
Thermal, Image

Biometric Sensors
Fingerprint, Heart rate, etc.

Force Sensors
Pressure, Strain

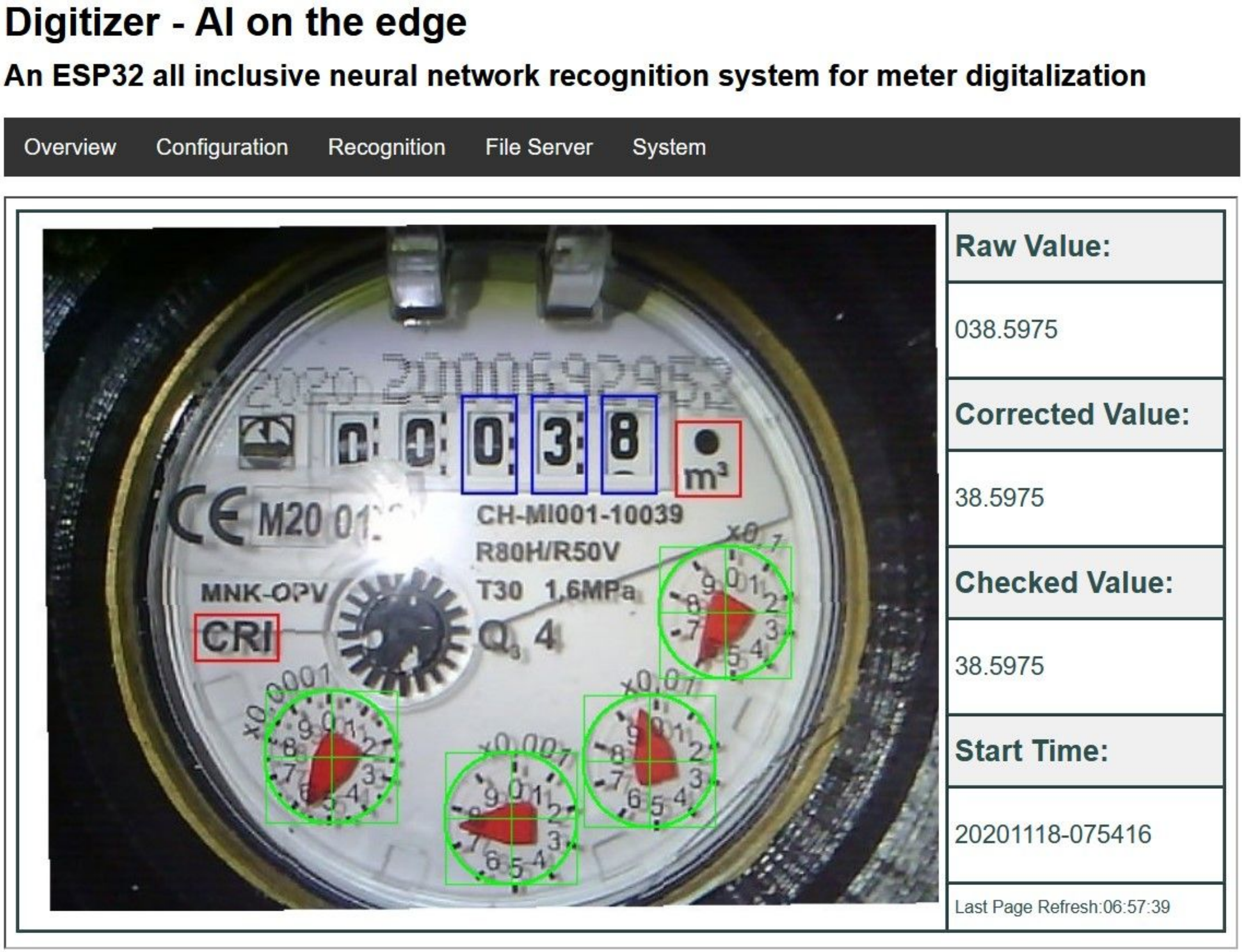
Rotation Sensors
Encoders

...



A DIGITAL "SMART" METER

AN ELECTROMECHANICAL ANALOG METER

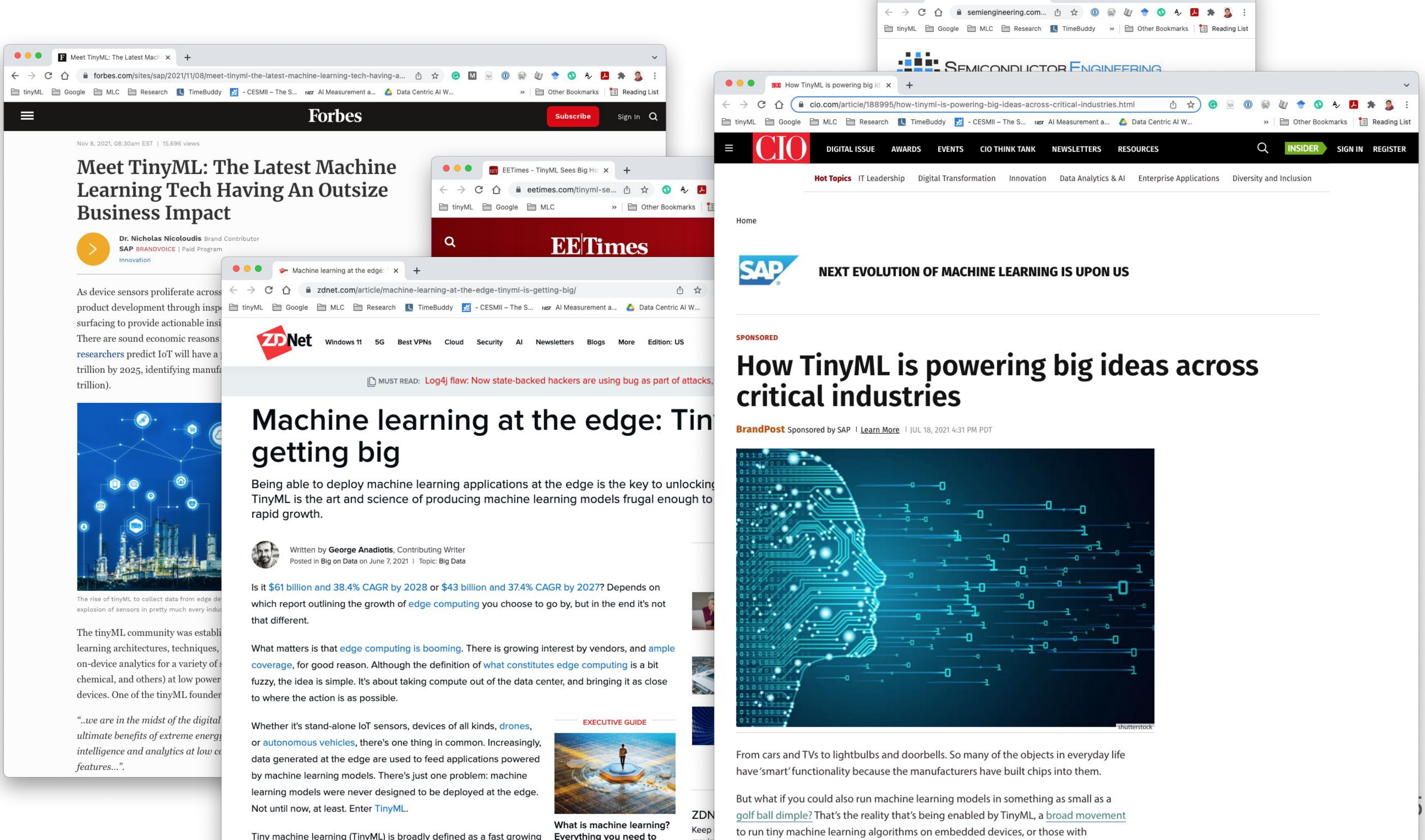


Digitizer - AI on the edge

An ESP32 all inclusive neural network recognition system for meter digitalization

Raw Value:	038.5975
Corrected Value:	38.5975
Checked Value:	38.5975
Start Time:	20201118-075416

Last Page Refresh:06:57:39


semiengineering.com...
tinyML Google MLC Research TimeBuddy Other Bookmarks Reading List
SEMICONDUCTOR ENGINEERING
Meet TinyML: The Latest Mach
forbes.com/sites/sap/2021/11/08/meet-tinyml-the-latest-machine-learning-tech-having-a...
tinyML Google MLC Research TimeBuddy - CESMII – The S... AI Measurement a... Data Centric AI W... Other Bookmarks Reading List
Forbes
Subscribe
Sign In
EETimes - TinyML Sees Big Ho
eetimes.com/tinyml-se...
tinyML Google MLC Other Bookmarks
EETimes
Machine learning at the edge:
zdnet.com/article/machine-learning-at-the-edge-tinyml-is-getting-big/
tinyML Google MLC Research TimeBuddy - CESMII – The S... AI Measurement a... Data Centric AI W...
ZDNet
Windows 11 5G Best VPNs Cloud Security AI Newsletters Blogs More Edition: US
How TinyML is powering big id
cio.com/article/188995/how-tinyml-is-powering-big-ideas-across-critical-industries.html
tinyML Google MLC Research TimeBuddy - CESMII – The S... AI Measurement a... Data Centric AI W... Other Bookmarks Reading List
CIO
DIGITAL ISSUE AWARDS EVENTS CIO THINK TANK NEWSLETTERS RESOURCES
INSIDER SIGN IN REGISTER
Hot Topics IT Leadership Digital Transformation Innovation Data Analytics & AI Enterprise Applications Diversity and Inclusion
shutterstock

Nov 8, 2021, 08:30am EST | 15,696 views

Meet TinyML: The Latest Machine Learning Tech Having An Outsize Business Impact

Dr. Nicholas Nicoloudis Brand Contributor
SAP BRANDVOICE | Paid Program
Innovation

As device sensors proliferate across
product development through insp
surfacing to provide actionable insi
There are sound economic reasons
researchers predict IoT will have a
trillion by 2025, identifying manuf
trillion).

The rise of tinyML to collect data from edge de
explosion of sensors in pretty much every indu

The tinyML community was establi
learning architectures, techniques,
on-device analytics for a variety of
chemical, and others) at low power
devices. One of the tinyML founder

"..we are in the midst of the digital
ultimate benefits of extreme energy
intelligence and analytics at low co
features…".

MUST READ: Log4j flaw: Now state-backed hackers are using bug as part of attacks,

Machine learning at the edge: Tin getting big

Being able to deploy machine learning applications at the edge is the key to unlocking
TinyML is the art and science of producing machine learning models frugal enough to
rapid growth.

Written by George Anadiotis, Contributing Writer
Posted in Big on Data on June 7, 2021 | Topic: Big Data

Is it $61 billion and 38.4% CAGR by 2028 or $43 billion and 37.4% CAGR by 2027? Depends on which report outlining the growth of edge computing you choose to go by, but in the end it's not that different.

What matters is that edge computing is booming. There is growing interest by vendors, and ample coverage, for good reason. Although the definition of what constitutes edge computing is a bit fuzzy, the idea is simple. It's about taking compute out of the data center, and bringing it as close to where the action is as possible.

Whether it's stand-alone IoT sensors, devices of all kinds, drones, or autonomous vehicles, there's one thing in common. Increasingly, data generated at the edge are used to feed applications powered by machine learning models. There's just one problem: machine learning models were never designed to be deployed at the edge. Not until now, at least. Enter TinyML.

EXECUTIVE GUIDE

What is machine learning? Everything you need to

Tiny machine learning (TinyML) is broadly defined as a fast growing

ZDN
Keep

Home

NEXT EVOLUTION OF MACHINE LEARNING IS UPON US

SPONSORED

How TinyML is powering big ideas across critical industries

BrandPost Sponsored by SAP | Learn More | JUL 18, 2021 4:31 PM PDT

From cars and TVs to lightbulbs and doorbells. So many of the objects in everyday life have 'smart' functionality because the manufacturers have built chips into them.

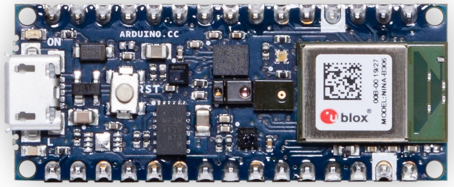
But what if you could also run machine learning models in something as small as a golf ball dimple? That's the reality that's being enabled by TinyML, a broad movement to run tiny machine learning algorithms on embedded devices, or those with

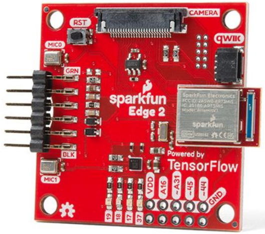
Challenges

250 Billion

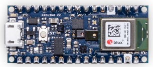
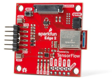
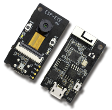
MCUs today

Himax

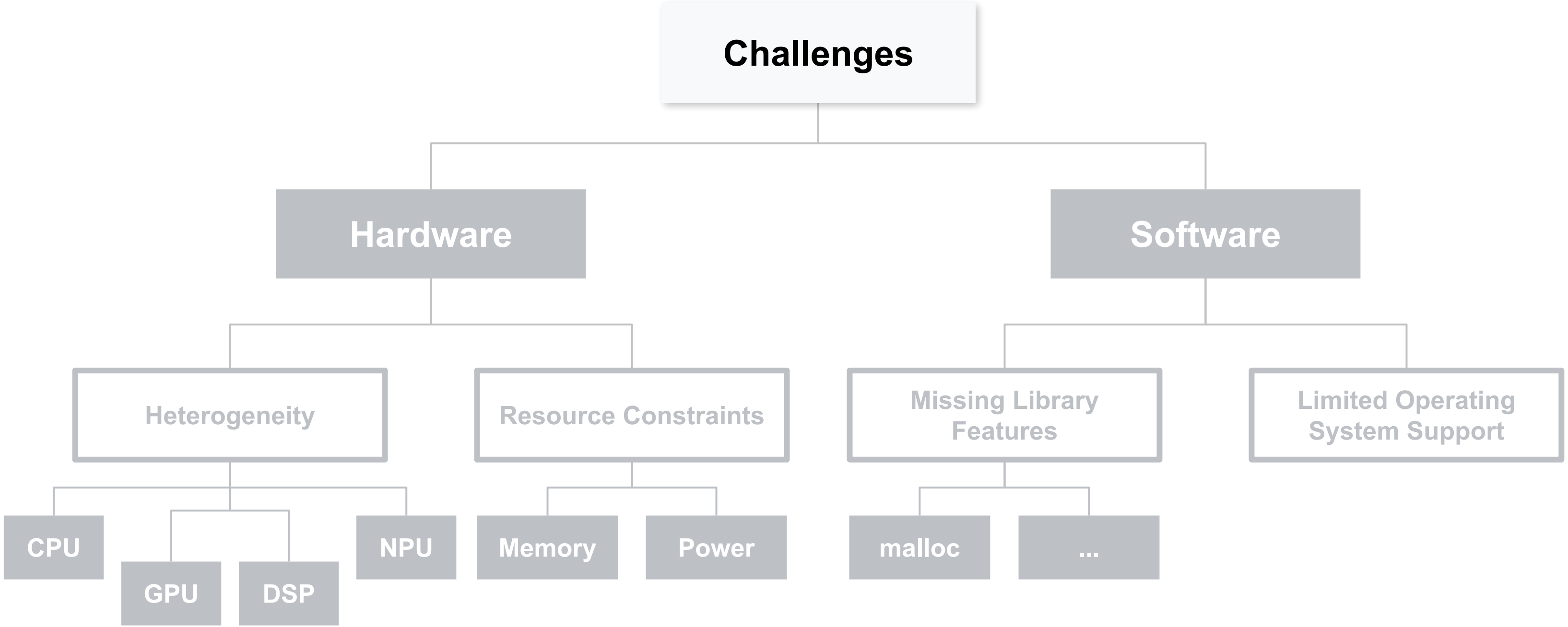

ARDUINO.CC
ON
RST
blox

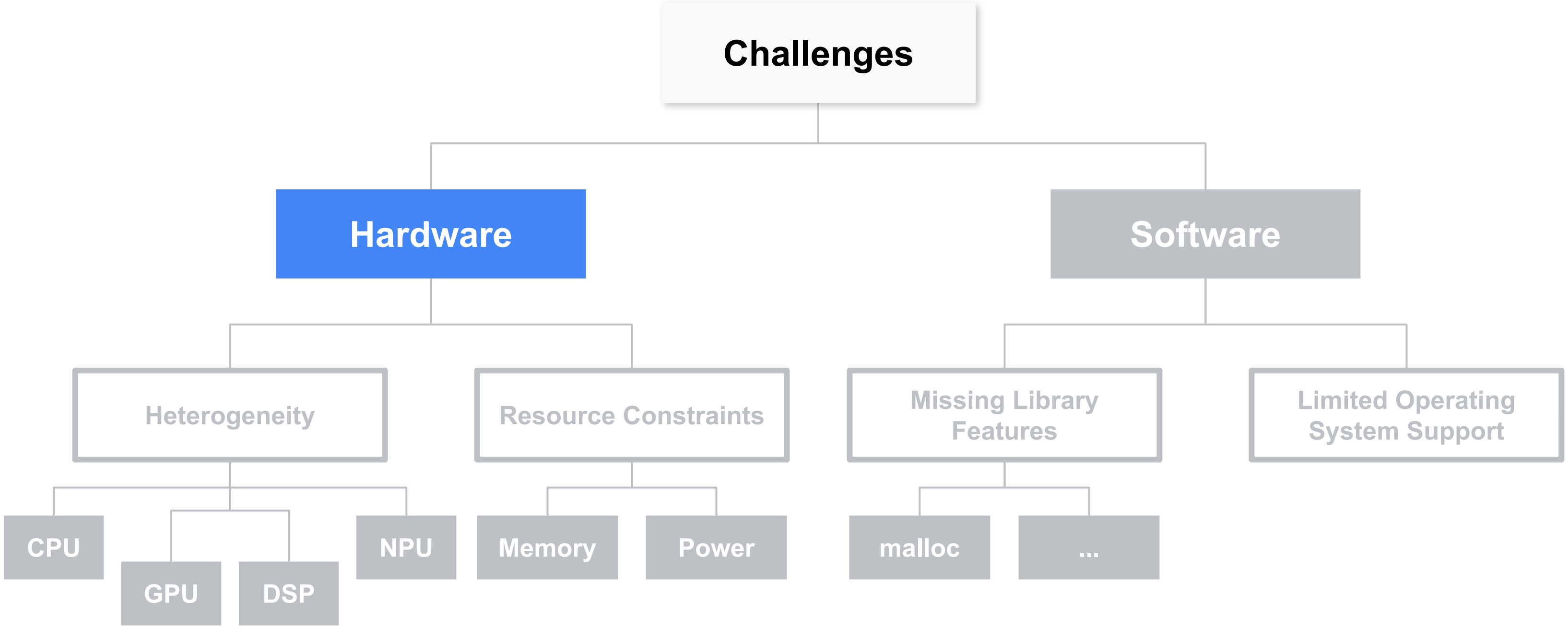

CAMERA
RST
qwiic
sparkfun
Edge 2
Powered by
TensorFlow

ESP-EYE
V2.1

	Board	MCU / ASIC	Clock	Memory	Sensors	Radio
	Himax WE-I Plus EVB	HX6537-A 32-bit EM9D DSP	400 MHz	2MB flash 2MB RAM	Accelerometer, Mic, Camera	None
	Arduino Nano 33 BLE Sense	32-bit nRF52840	64 MHz	1MB flash 256kB RAM	Mic, IMU, Temp, Humidity, Gesture, Pressure, Proximity, Brightness, Color	BLE
	SparkFun Edge 2	32-bit ArtemisV1	48 MHz	1MB flash 384kB RAM	Accelerometer, Mic, Camera	BLE
	Espressif EYE	32-bit ESP32-D0WD	240 MHz	4MB flash 520kB RAM	Mic, Camera	WiFi, BLE

Challenges


Challenges
Hardware
Software
Heterogeneity
Resource Constraints
Missing Library Features
Limited Operating System Support
CPU
GPU
DSP
NPU
Memory
Power
malloc
...



Challenges

Hardware

Software

Heterogeneity

Resource Constraints

Missing Library Features

Limited Operating System Support

CPU

GPU

DSP

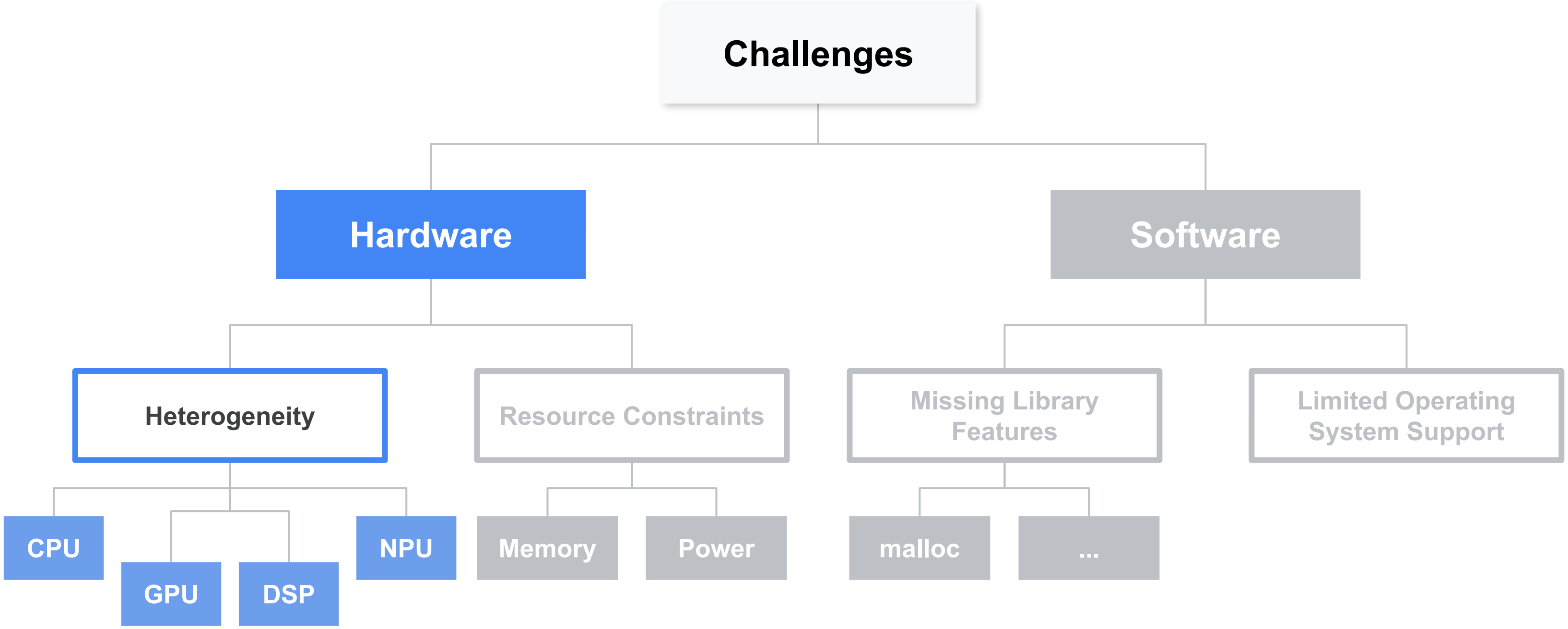
NPU

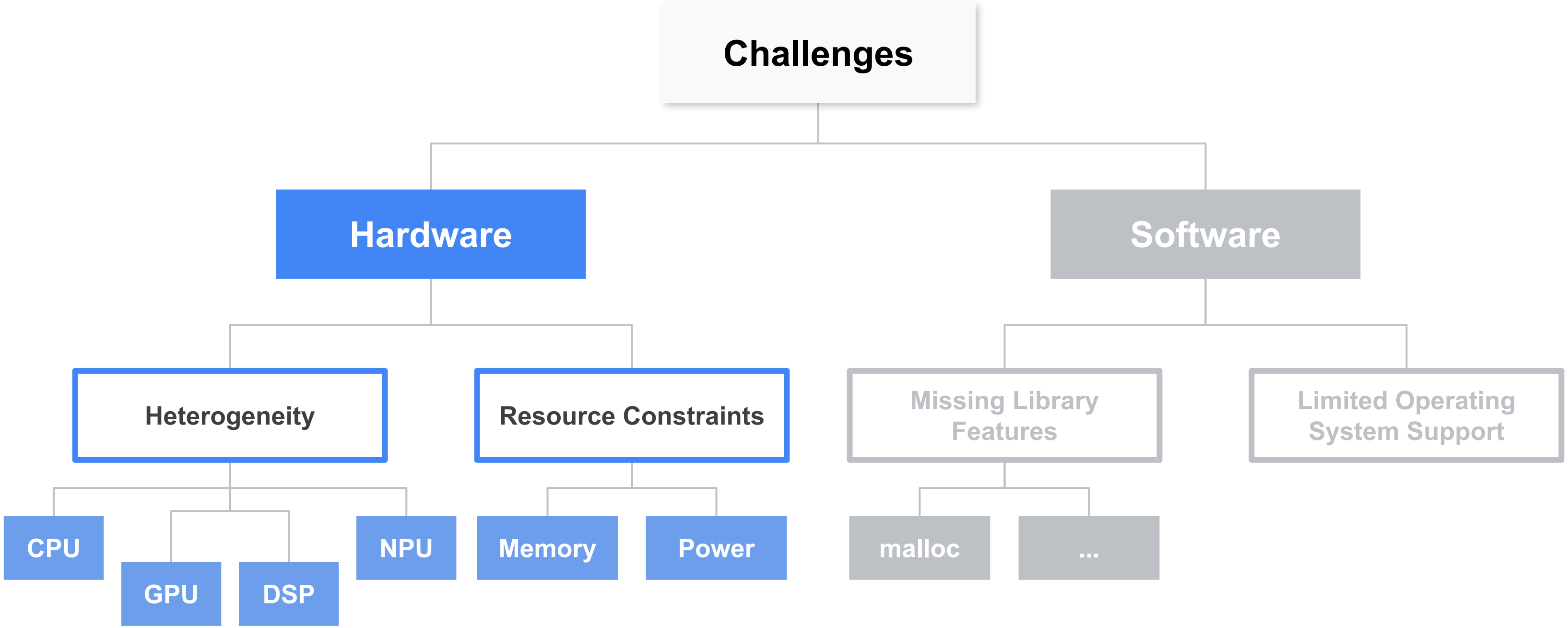
Memory

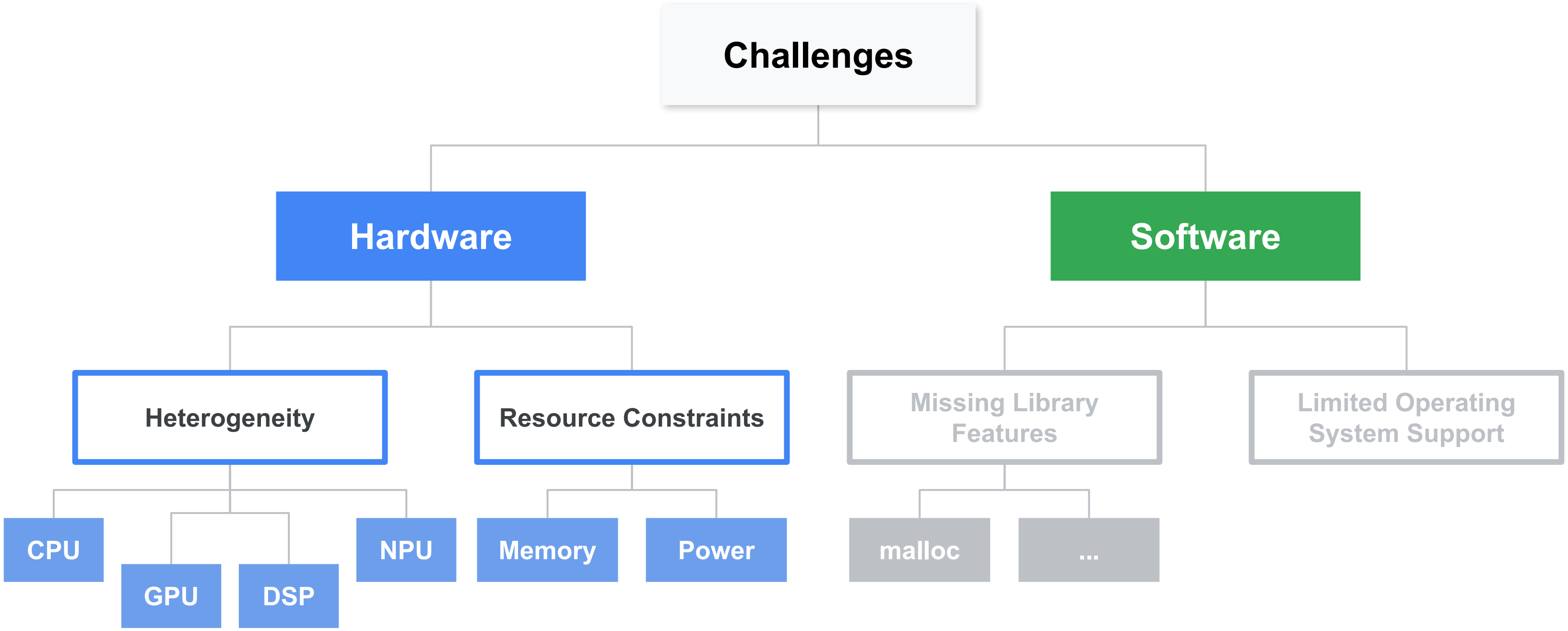
Power

malloc

...


Challenges
Hardware
Software
Heterogeneity
Resource Constraints
Missing Library Features
Limited Operating System Support
CPU
GPU
DSP
NPU
Memory
Power
malloc
...


Challenges
Hardware
Software
Heterogeneity
Resource Constraints
Missing Library Features
Limited Operating System Support
CPU
GPU
DSP
NPU
Memory
Power
malloc
...



Challenges

Hardware

Software

Heterogeneity

Resource Constraints

Missing Library Features

Limited Operating System Support

CPU

GPU

DSP

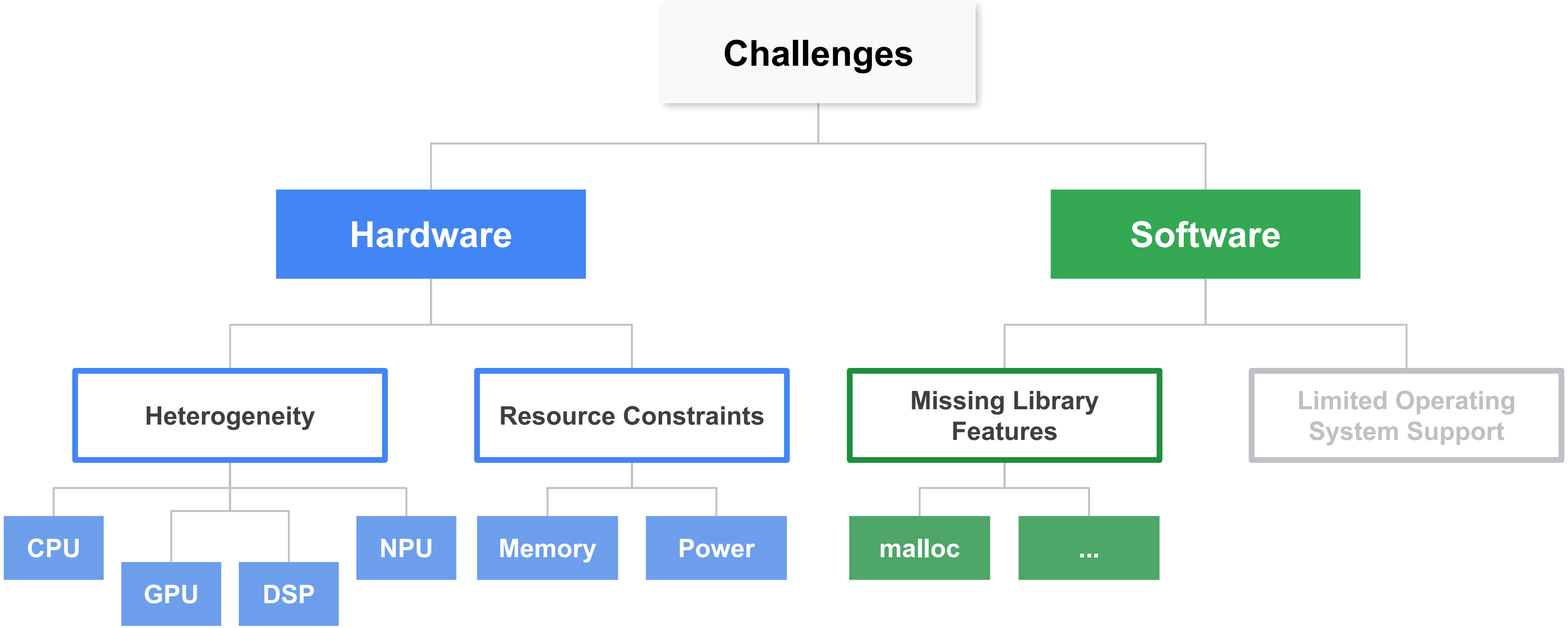
NPU

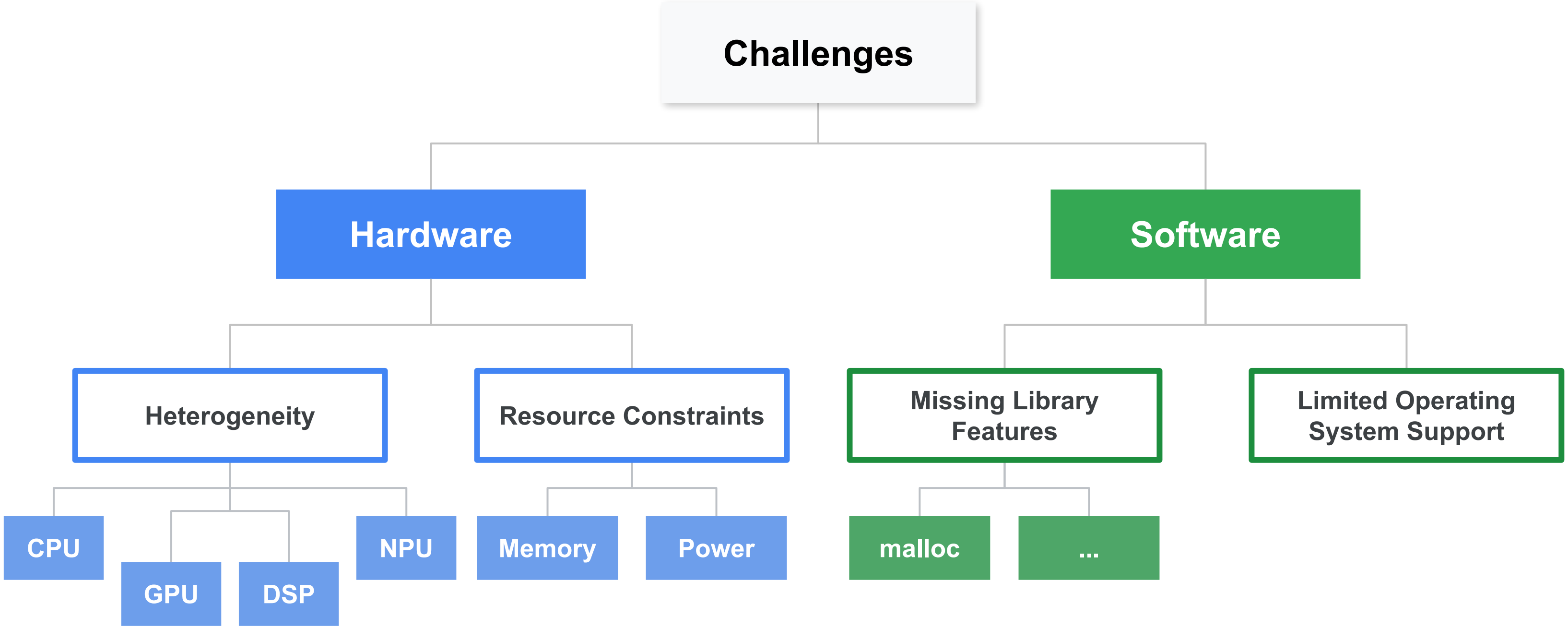
Memory

Power

malloc

...


Challenges
Hardware
Software
Heterogeneity
Resource Constraints
Missing Library Features
Limited Operating System Support
CPU
GPU
DSP
NPU
Memory
Power
malloc
...



Challenges

Hardware

Software

Heterogeneity

Resource Constraints

Missing Library Features

Limited Operating System Support

CPU

GPU

DSP

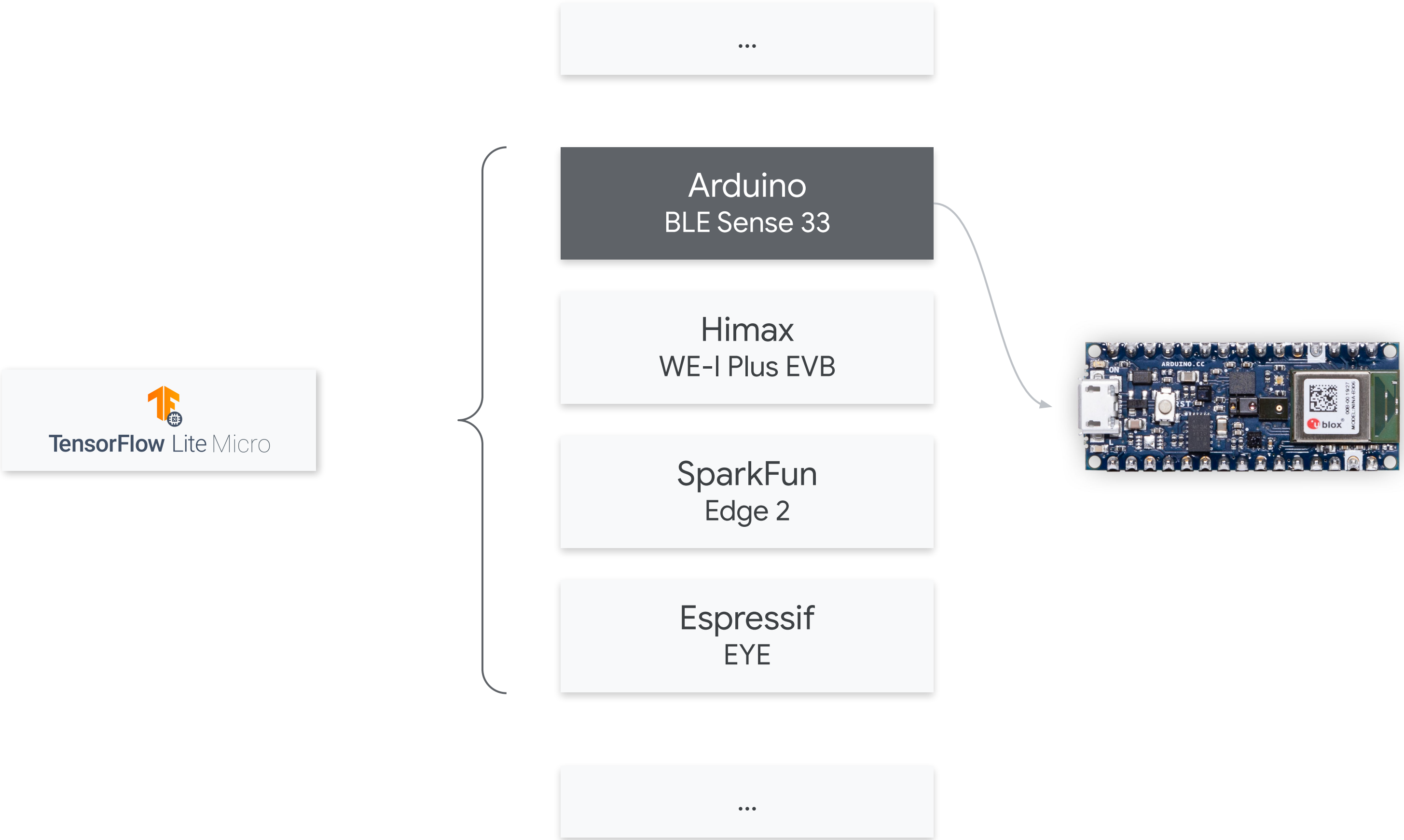
NPU

Memory

Power

malloc

...



TensorFlow Lite Micro

...

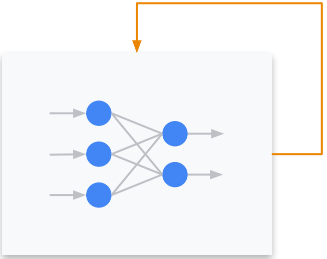
Arduino
BLE Sense 33

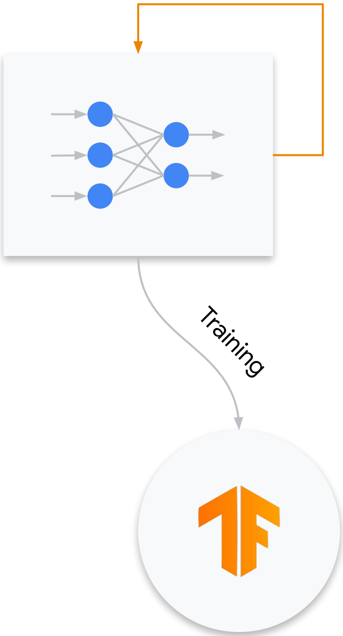
Himax
WE-I Plus EVB

SparkFun
Edge 2

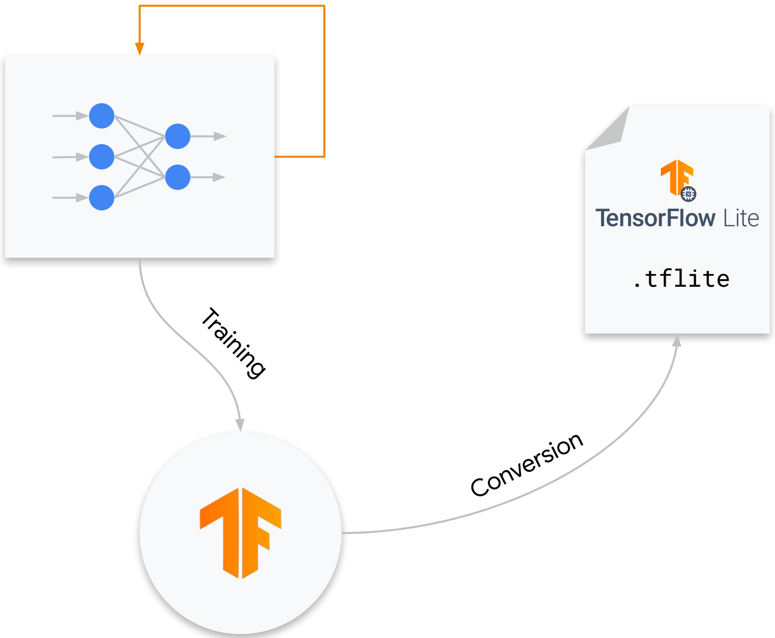
Espressif
EYE

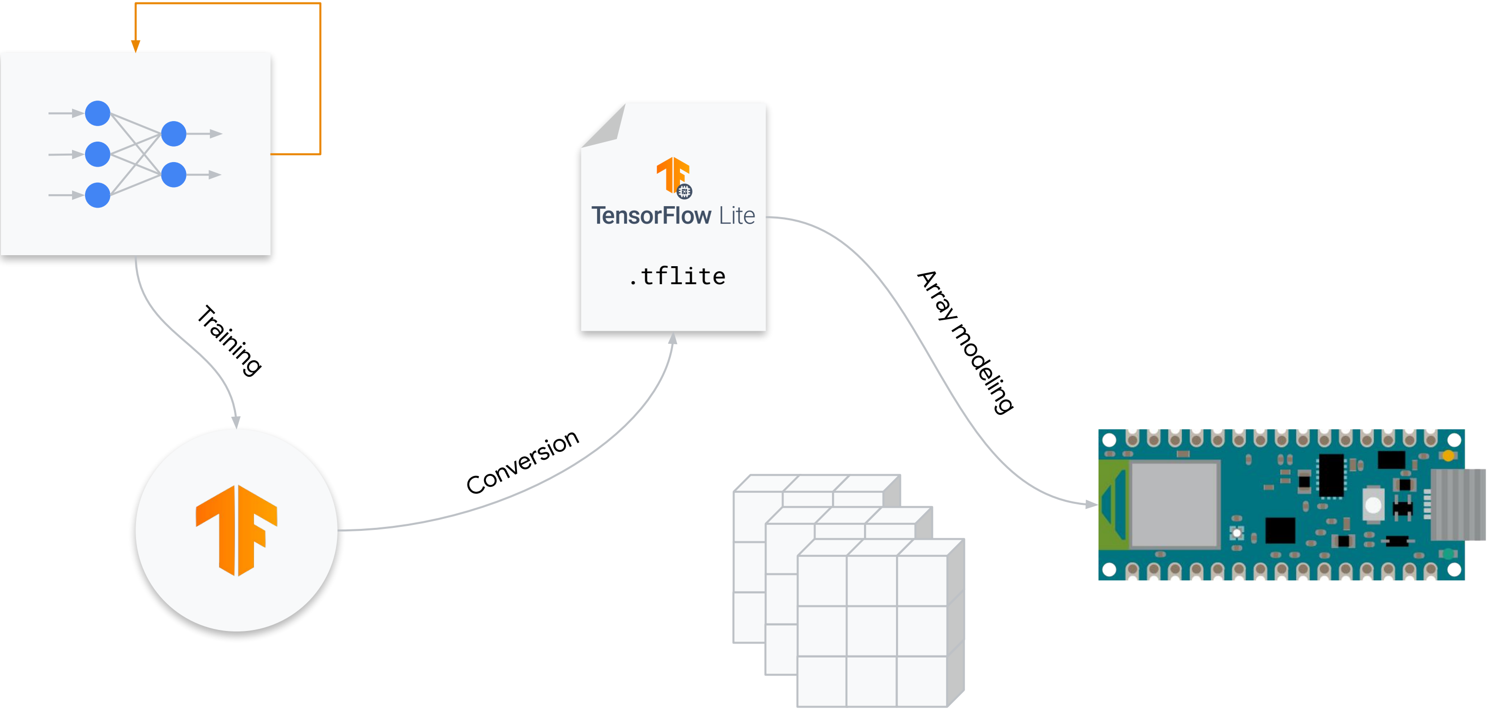
...





Training


Training
Conversion
TensorFlow Lite
.tflite



Training

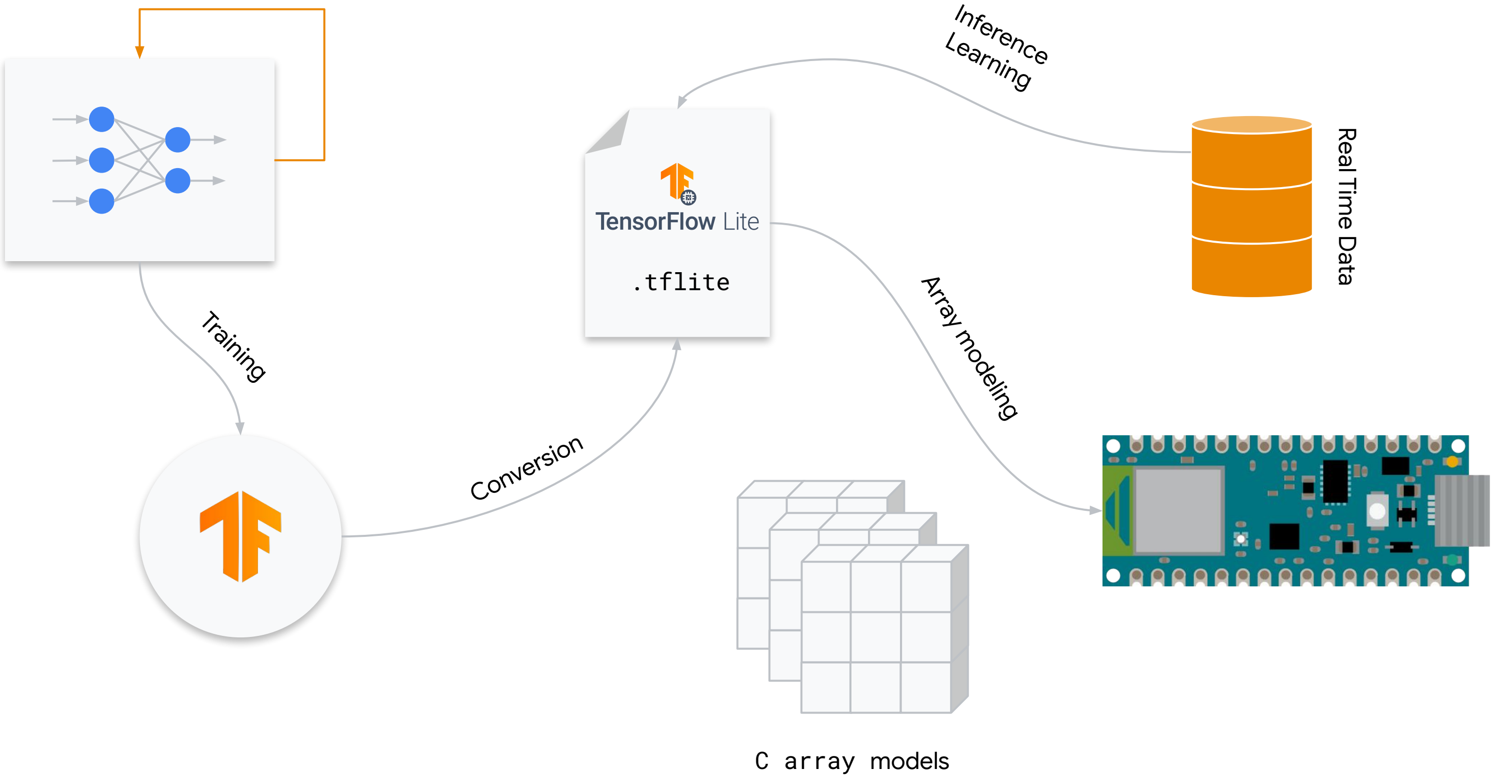
Conversion

TensorFlow Lite

.tflite

Array modeling

C array models



Training

Conversion

TensorFlow Lite

.tflite

Inference
Learning

Real Time Data

Array modeling

C array models

TensorFlow Lite Micro
in a Nutshell

Built to fit on **embedded systems**:

- Very **small binary footprint**
- **No** dynamic memory allocation
- **No** dependencies on complex parts of the standard C/C++ libraries
- **No** operating system dependencies, **can run on bare metal**
- Designed to be **portable** across a wide variety of systems

TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems

Robert David[1] Jared Duke[1] Advait Jain[1] Vijay Janapa Reddi[1 2]
Nat Jeffries[1] Jian Li[1] Nick Kreeger[1] Ian Nappier[1] Meghna Natraj[1]
Shlomi Regev[1] Rocky Rhodes[1] Tiezhen Wang[1] Pete Warden[1]

arXiv:2010.08678v3 [cs.LG] 13 Mar 2021

Abstract

TensorFlow Lite Micro (TFLM) is an open-source ML inference framework for running deep-learning models on embedded systems. TFLM tackles the efficiency requirements imposed by embedded-system resource constraints and the fragmentation challenges that make cross-platform interoperability nearly impossible. The framework adopts a unique interpreter-based approach that provides flexibility while overcoming these unique challenges. In this paper, we explain the design decisions behind TFLM and describe its implementation. We present an evaluation of TFLM to demonstrate its low resource requirements and minimal run-time performance overheads.

1 Introduction

Tiny machine learning (TinyML) is a burgeoning field at the intersection of embedded systems and machine learning. The world has over 250 billion microcontrollers (IC Insights, 2020), with strong growth projected over coming years. As such, a new range of embedded applications are emerging for neural networks. Because these models are extremely small (few hundred KBs), running on microcontrollers or DSP-based embedded subsystems, they can operate continuously with minimal impact on device battery life.

The most well-known and widely deployed example of this new TinyML technology is keyword spotting, also called hotword or wakeword detection (Chen et al., 2014; Gruenstein et al., 2017; Zhang et al., 2017). Amazon, Apple, Google, and others use tiny neural networks on billions of devices to run always-on inferences for keyword detection—and this is far from the only TinyML application. Low-latency analysis and modeling of sensor signals from microphones, low-power image sensors, accelerometers, gyros, PPG optical sensors, and other devices enable consumer and industrial applications, including predictive maintenance (Goebel et al., 2020; Susto et al., 2014), acoustic-anomaly detection (Koizumi et al., 2019), visual object detection (Chowdhery et al., 2019), and human-activity recognition (Chavarriaga et al., 2013; Zhang & Sawchuk, 2012).

Unlocking machine learning's potential in embedded devices requires overcoming two crucial challenges. First and foremost, embedded systems have no unified TinyML framework. When engineers have deployed neural networks to such systems, they have built one-off frameworks that require manual optimization for each hardware platform. Such custom frameworks have tended to be narrowly focused, lacking features to support multiple applications and lacking portability across a wide range of hardware. The developer experience has therefore been painful, requiring hand optimization of models to run on a specific device. And altering these models to run on another device necessitated manual porting and repeated optimization effort. An important second-order effect of this situation is that the slow pace and high cost of training and deploying models to embedded hardware prevents developers from easily justifying the investment required to build new features.

Another challenge limiting TinyML is that hardware vendors have related but separate needs. Without a generic TinyML framework, evaluating hardware performance in a neutral, vendor-agnostic manner has been difficult. Frameworks are tied to specific devices, and it is hard to determine the source of improvements because they can come from hardware, software, or the complete vertically integrated solution.

The lack of a proper framework has been a barrier to accelerating TinyML adoption and application in products. Beyond deploying a model to an embedded target, the framework must also have a means of training a model on a higher-compute platform. TinyML must exploit a broad ecosystem of tools for ML, as well for orchestrating and debugging models, which are beneficial for production devices.

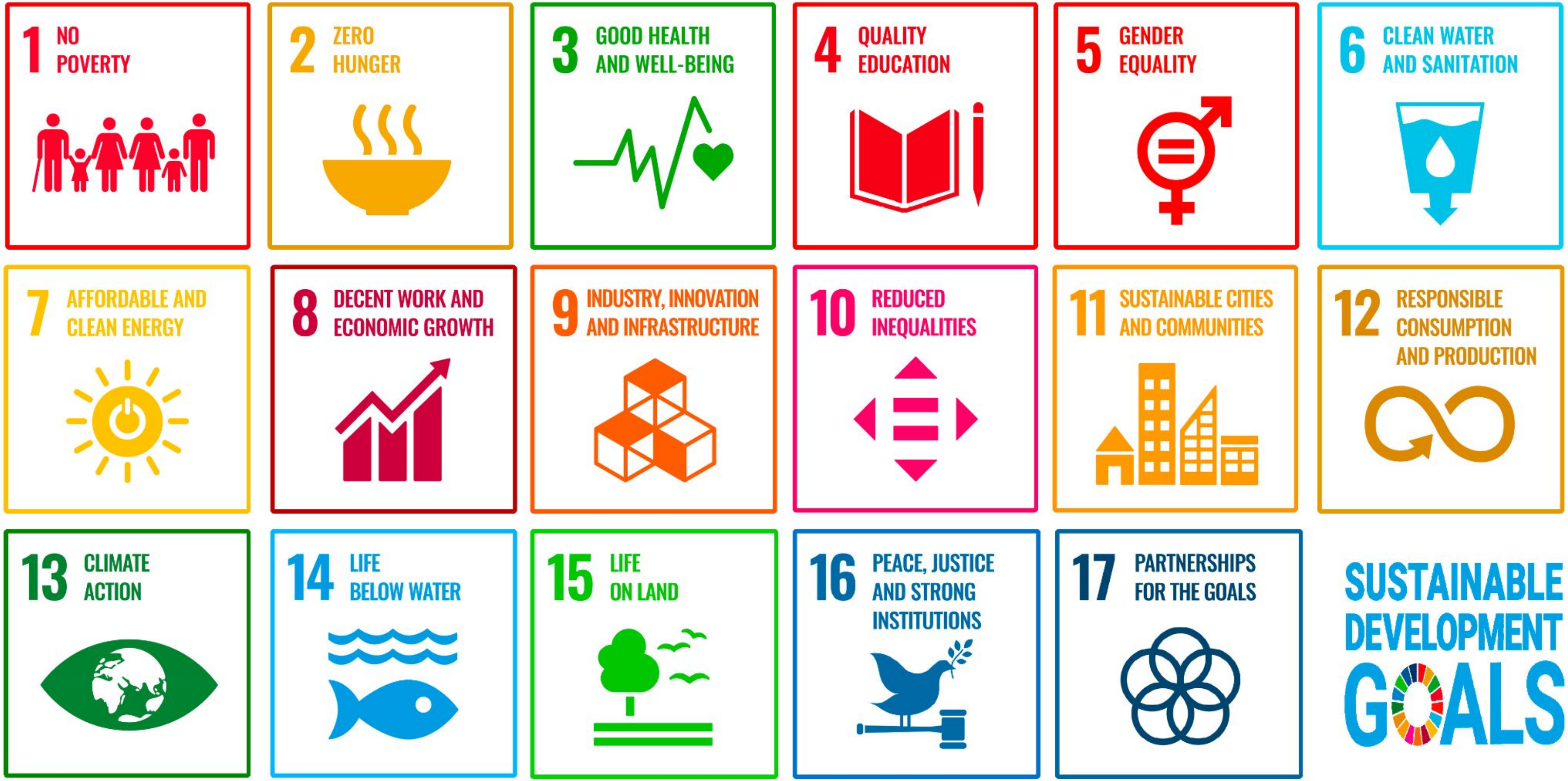
Prior efforts have attempted to bridge this gap. We can distill the major issues facing the frameworks into the following:

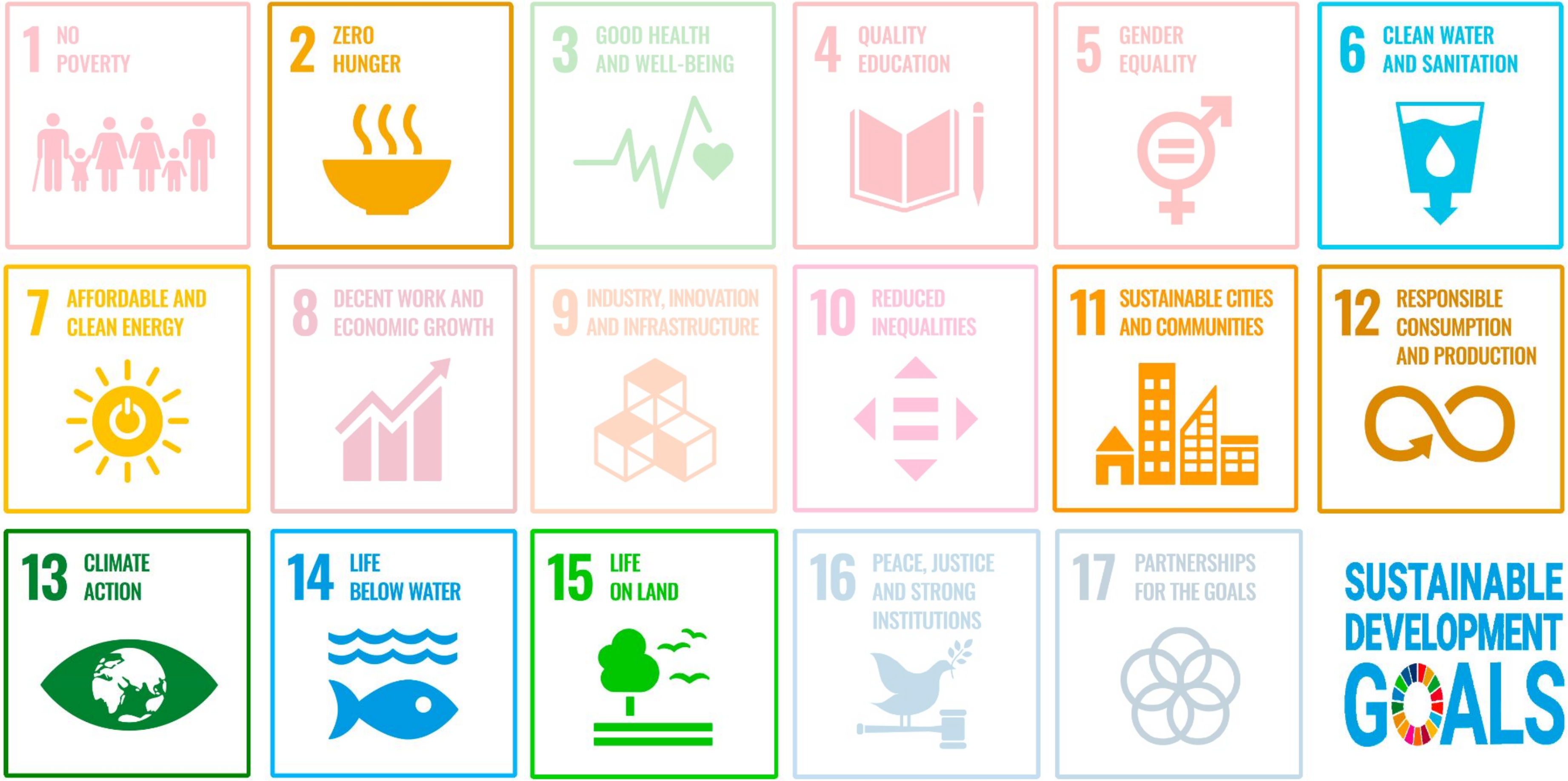
[1] Google [2] Harvard University. Correspondence to: Pete Warden <petewarden@google.com>, Vijay Janapa Reddi <vj@eecs.harvard.edu>.

Proceedings of the 4th MLSys Conference, San Jose, CA, USA, 2021. Copyright 2021 by the author(s).

[MLSys'21]

A Greener Tomorrow with TinyML


1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS
SUSTAINABLE DEVELOPMENT GOALS


1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS
SUSTAINABLE DEVELOPMENT GOALS

Sustainable TinyML

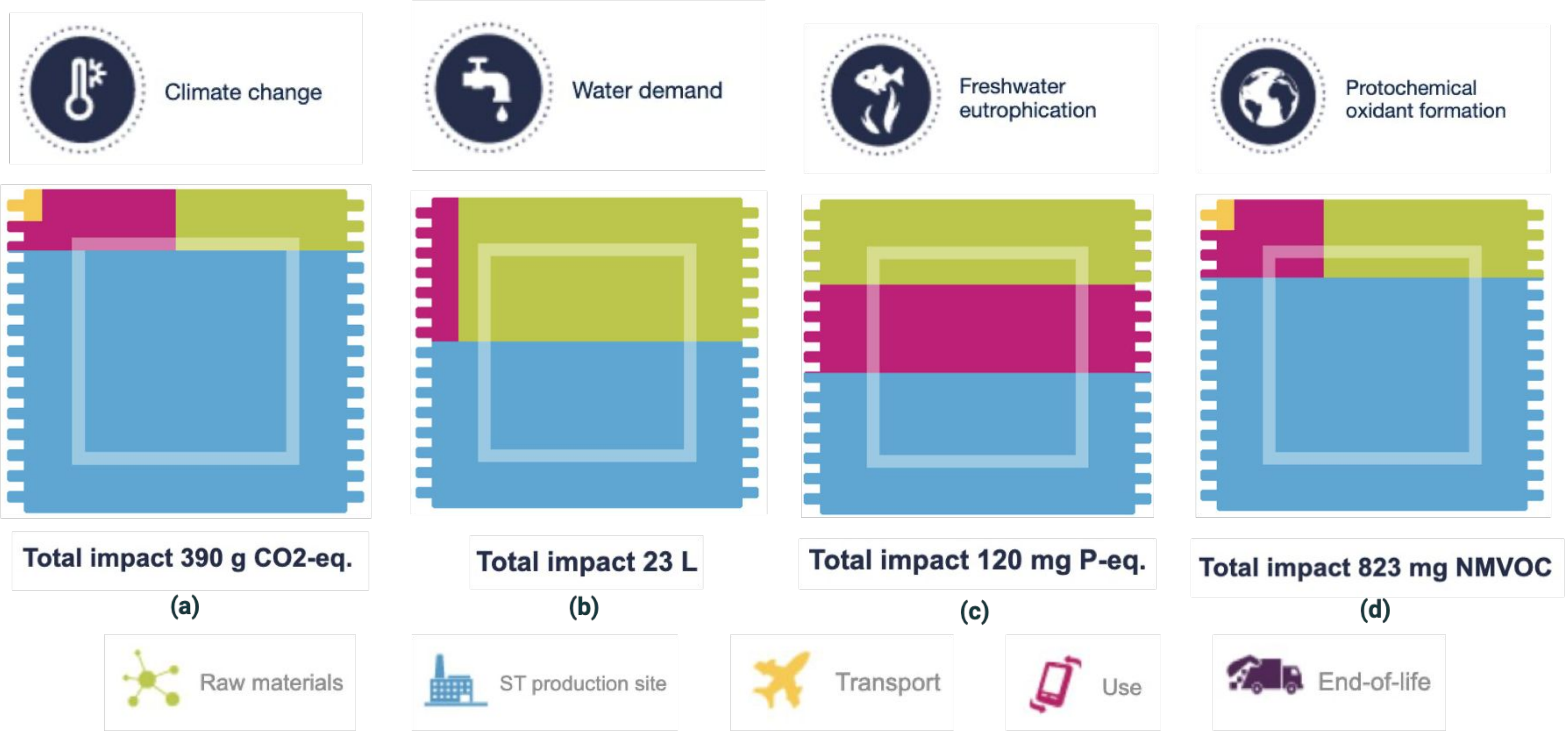
Climate change

Water demand

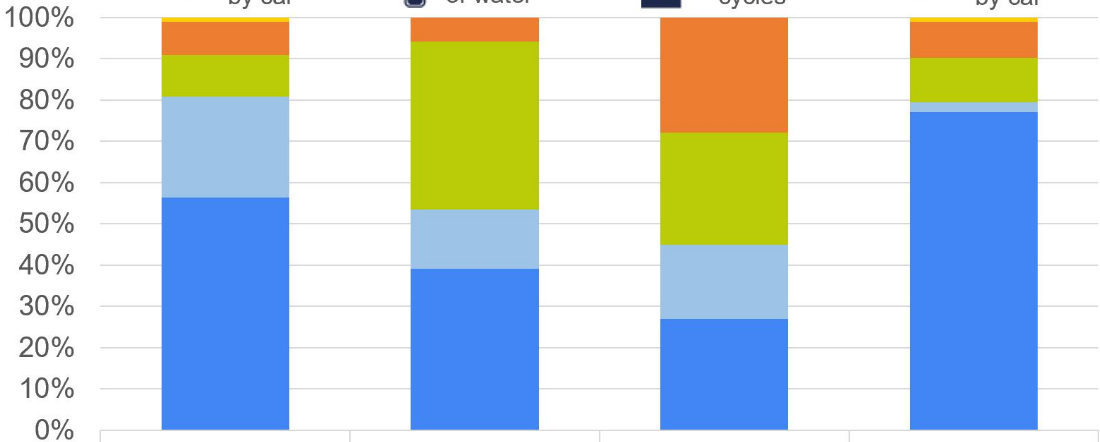
Freshwater eutrophication

Protochemical oxidant formation

Sustainable TinyML

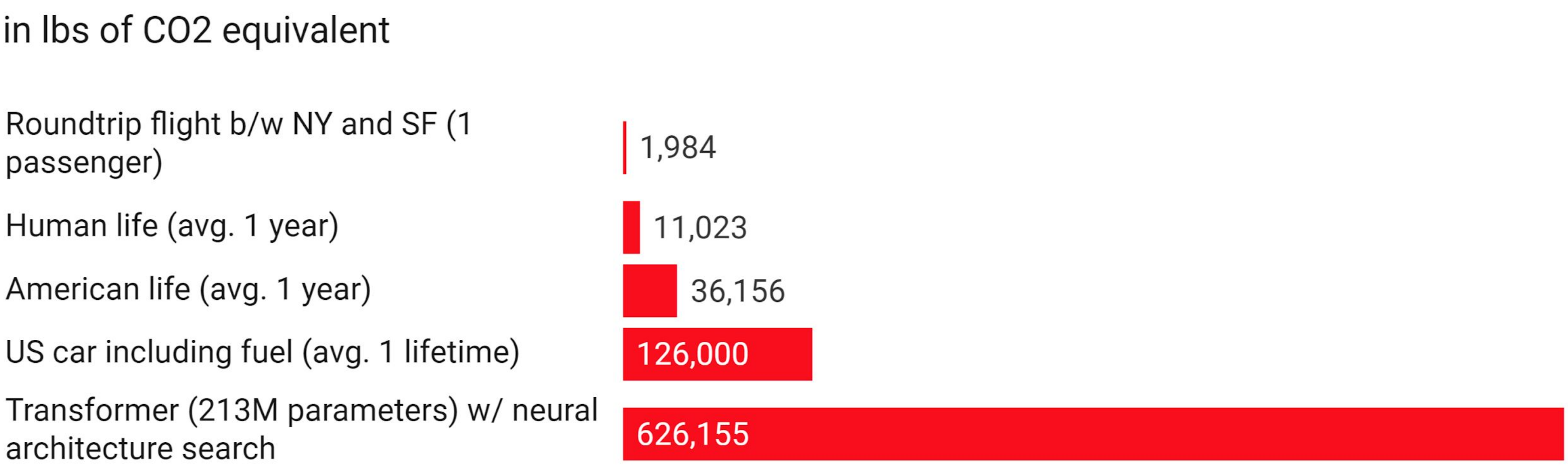


Tiny Footprint of a Microcontroller



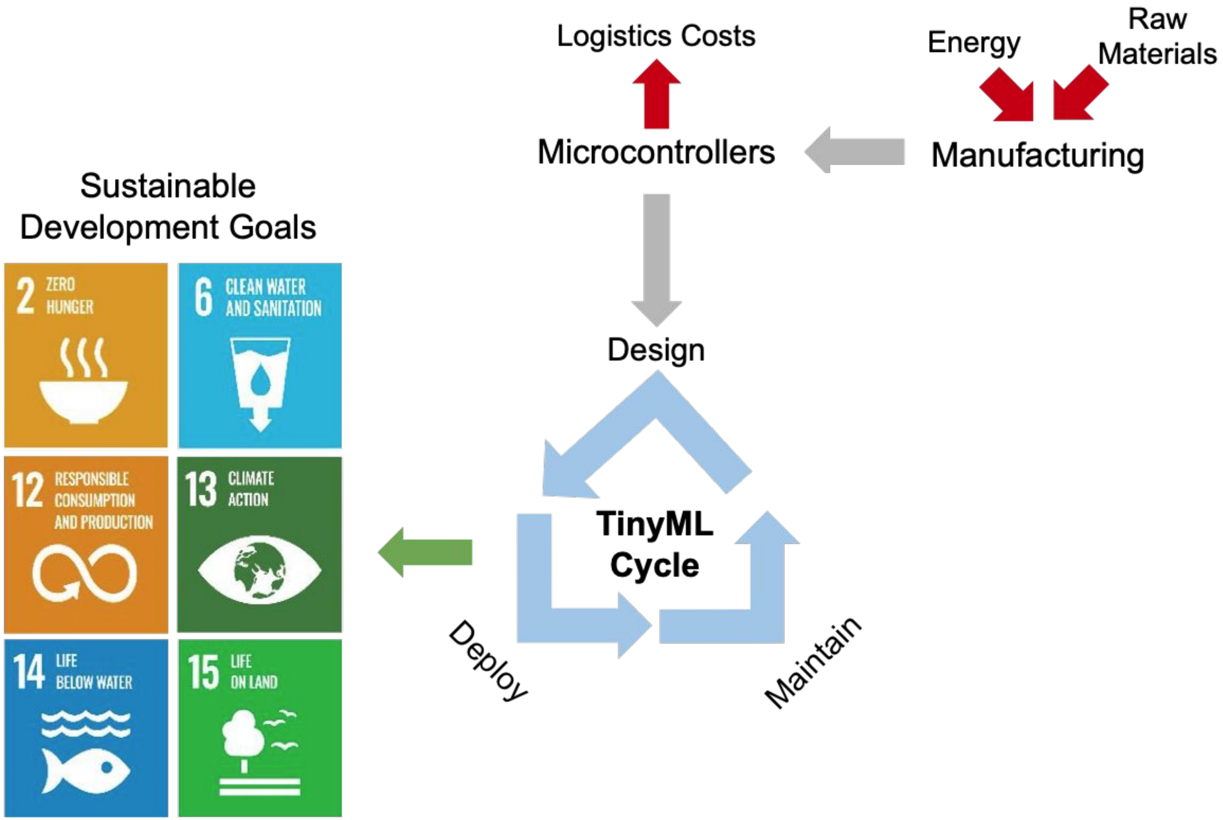
	Climate Change	Water Demand	Freshwater Eutrophication	Protochemical Oxidant Formation
Total Impact	390g CO_2-eq 1.6km by car	23L 23 bottles of water	120mg P-eq 0.2 washing cycles	823mg NMVOC 2.7km by car
🟪 End of Life	<1%	<1%	<1%	<1%
🟨 Logistics	1%	<1%	<1%	1%
🟧 Use	8%	6%	28%	8%
🟩 Raw Materials	10%	41%	27%	10%
▧ Production: Other	24%	15%	18%	2%
🟦 Production: Energy Consumption	56%	39%	27%	71%

Common Carbon Footprint Benchmarks



in lbs of CO2 equivalent

Chart: MIT Technology Review • Source: Strubell et al. • Created with Datawrapper



Sustainable Development Goals

Logistics Costs

Energy

Raw Materials

Microcontrollers

Manufacturing

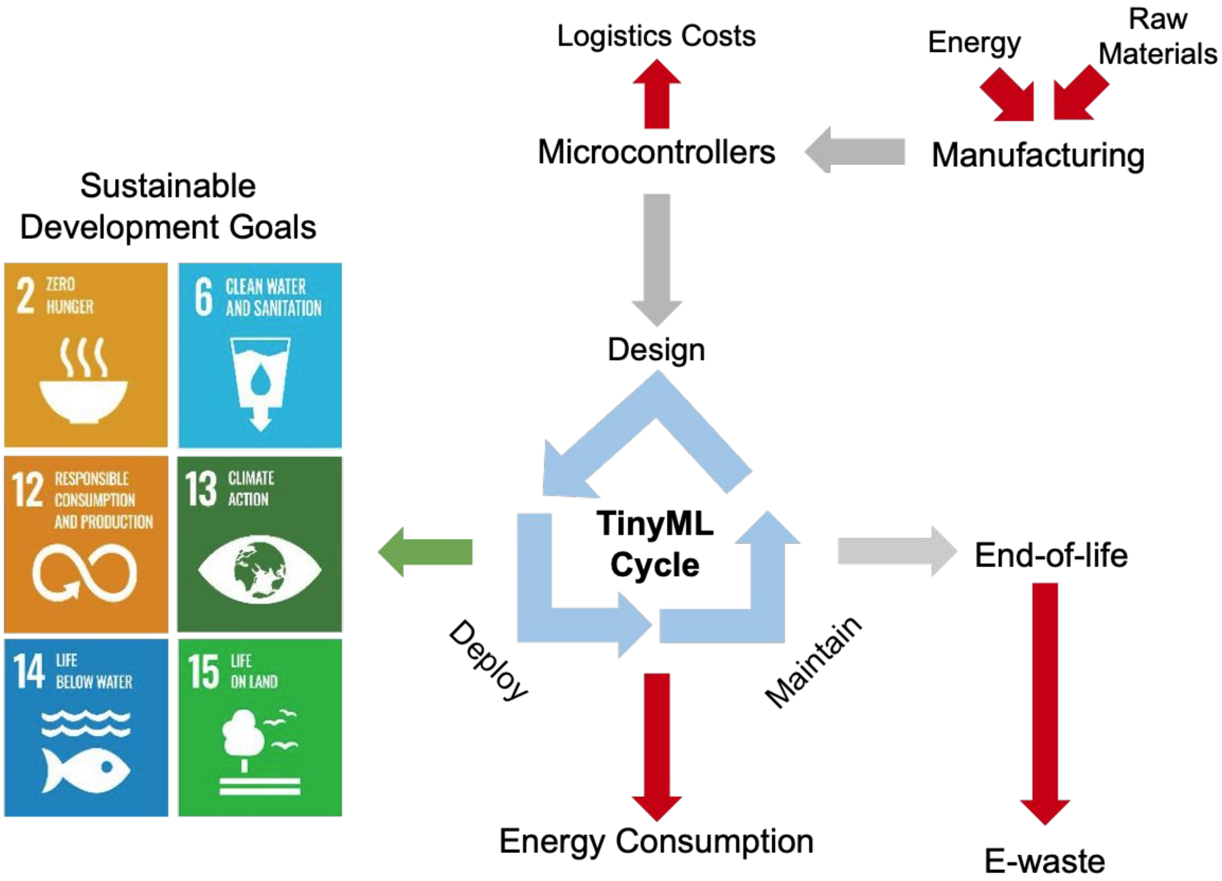
Design

TinyML Cycle

Deploy

Maintain

2 ZERO HUNGER
6 CLEAN WATER AND SANITATION
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND



Sustainable Development Goals

2 ZERO HUNGER
6 CLEAN WATER AND SANITATION
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND

Logistics Costs

Energy

Raw Materials

Microcontrollers

Manufacturing

Design

TinyML Cycle

Deploy

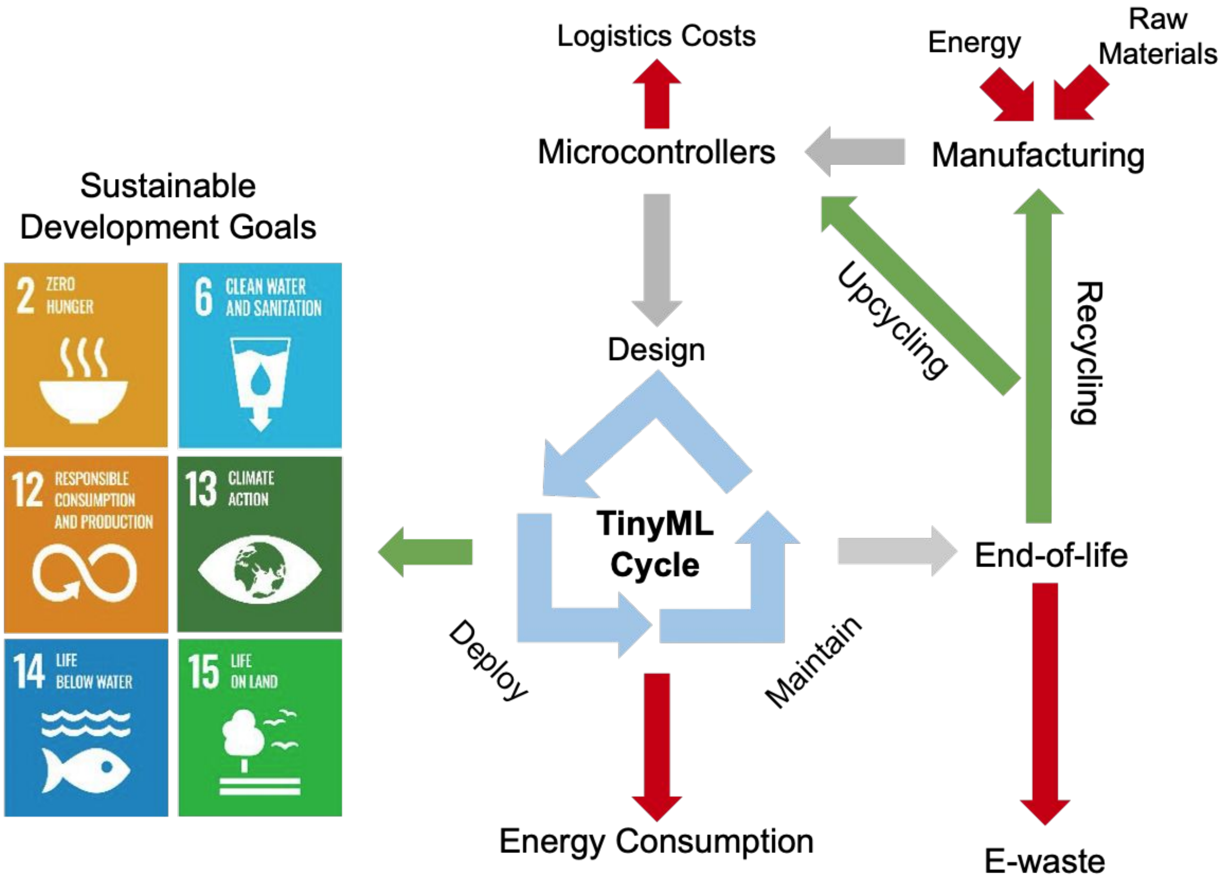
Maintain

End-of-life

Energy Consumption

E-waste

Sustainable Development Goals



2 ZERO HUNGER
6 CLEAN WATER AND SANITATION
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND

Logistics Costs

Energy

Raw Materials

Microcontrollers

Manufacturing

Design

Upcycling

Recycling

TinyML Cycle

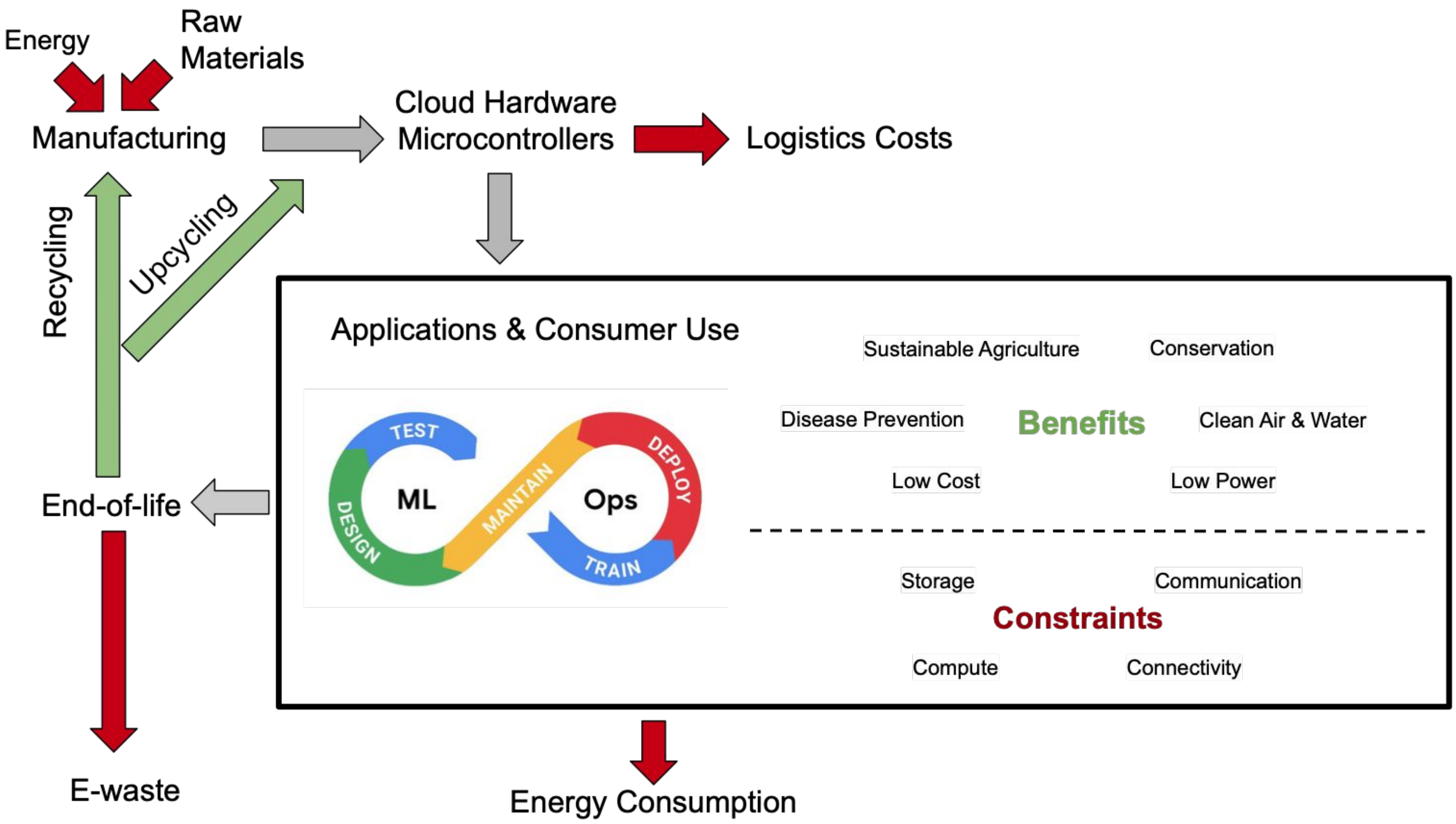
Deploy

Maintain

End-of-life

Energy Consumption

E-waste



Energy

Raw Materials

Manufacturing

Cloud Hardware Microcontrollers

Logistics Costs

Recycling

Upcycling

End-of-life

E-waste

Applications & Consumer Use

Sustainable Agriculture

Conservation

Disease Prevention

Benefits

Clean Air & Water

Low Cost

Low Power

Storage

Communication

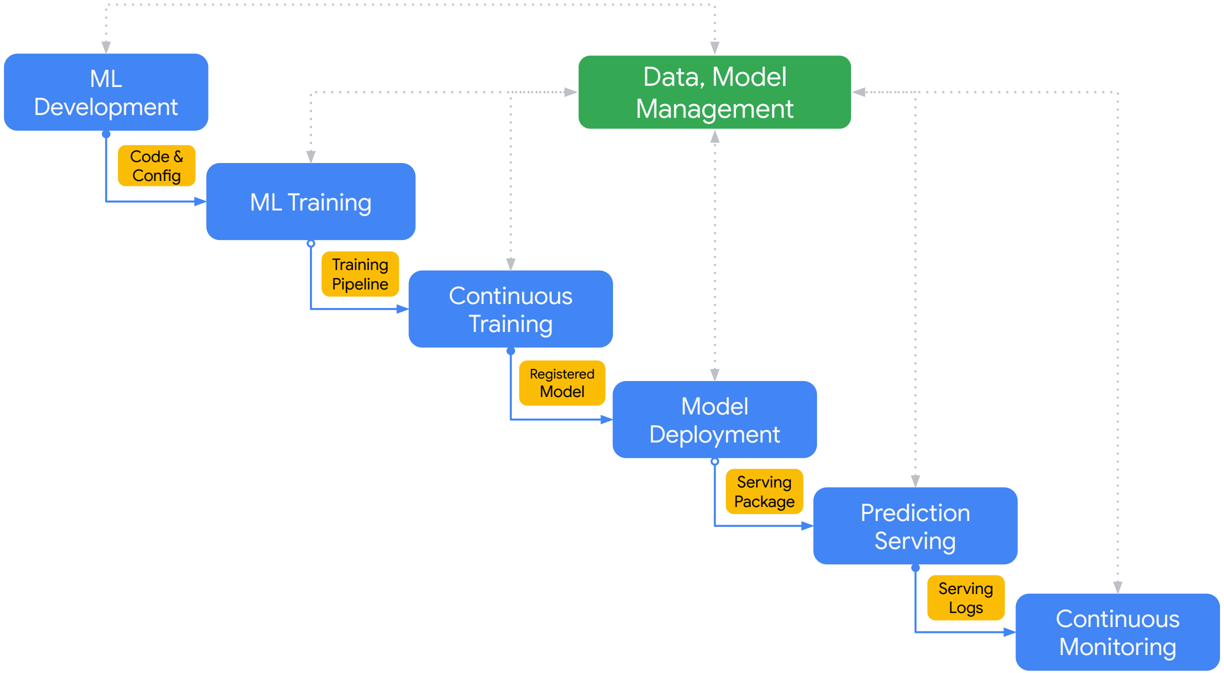
Constraints

Compute

Connectivity

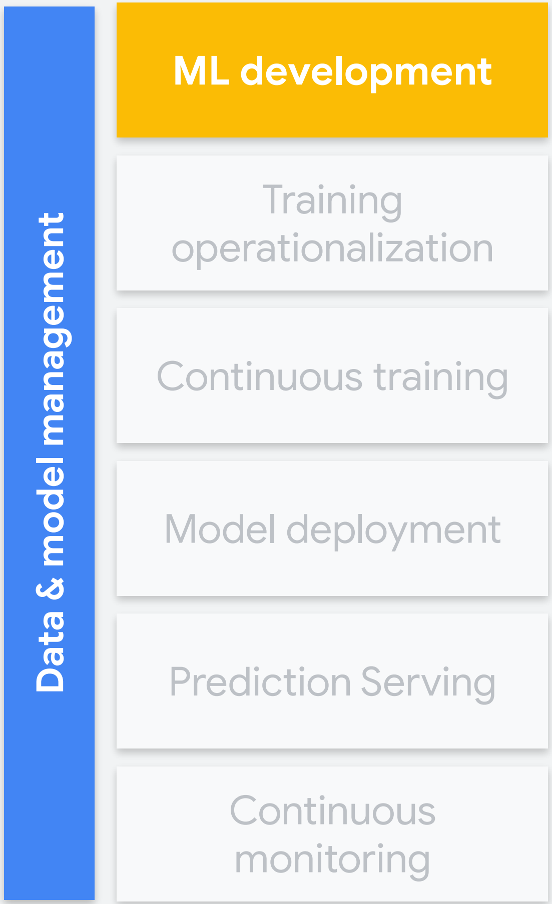
Energy Consumption

The MLOps Process



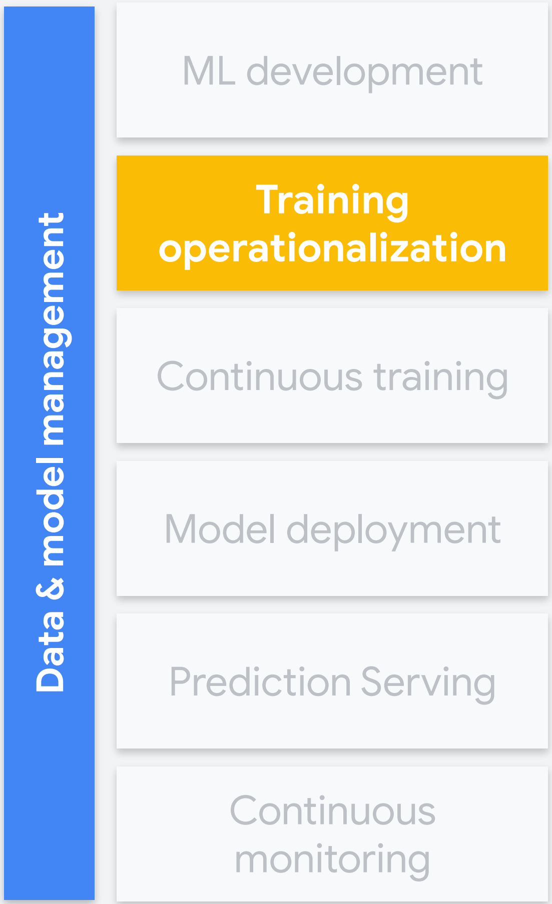
ML Development

ML development entails experimenting with and establishing a dependable and repeatable model training procedure.



Training Operationalization

Training operationalization is all about automating the packaging, testing, and deployment of repeatable and dependable training pipelines.



Data & model management

ML development

Training operationalization

Continuous training

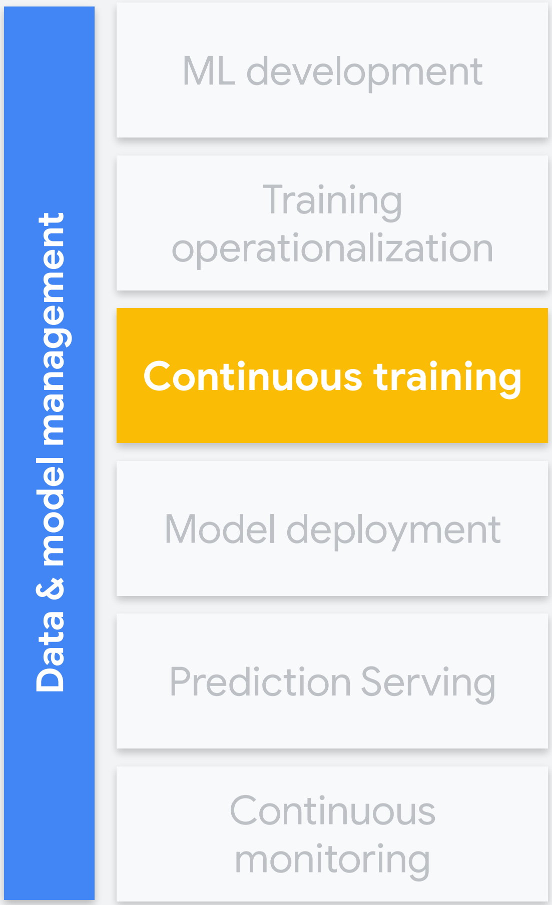
Model deployment

Prediction Serving

Continuous monitoring

Continuous Training

Continuous training entails running the training pipeline on a regular basis, maybe with fresh training settings, in response to new data or code modifications.



Data & model management

ML development

Training operationalization

Continuous training

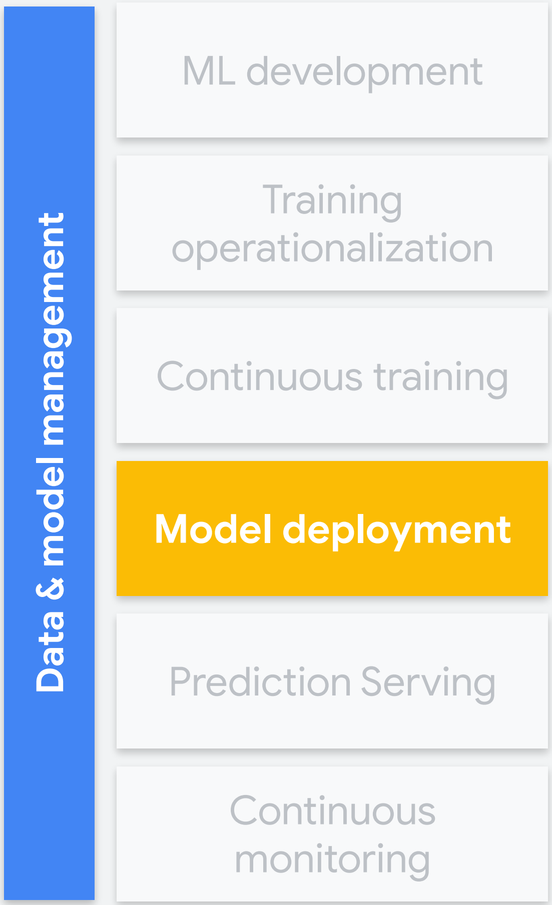
Model deployment

Prediction Serving

Continuous monitoring

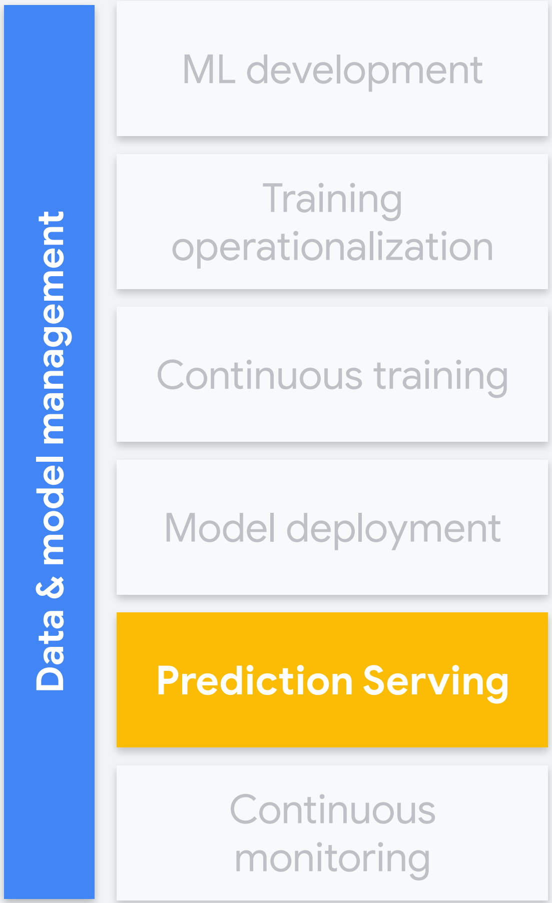
Model Deployment

Packaging, testing, and deploying a model to a serving environment for online experimentation and production serving is what model deployment is all about.



Prediction Serving

Serving the model that is deployed in production for inference is known as prediction serving.



Data & model management

ML development

Training operationalization

Continuous training

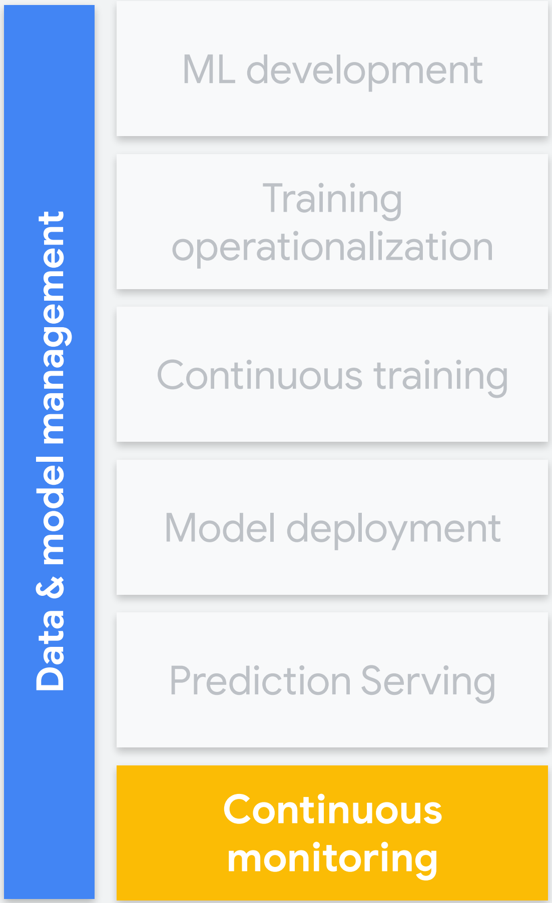
Model deployment

Prediction Serving

Continuous monitoring

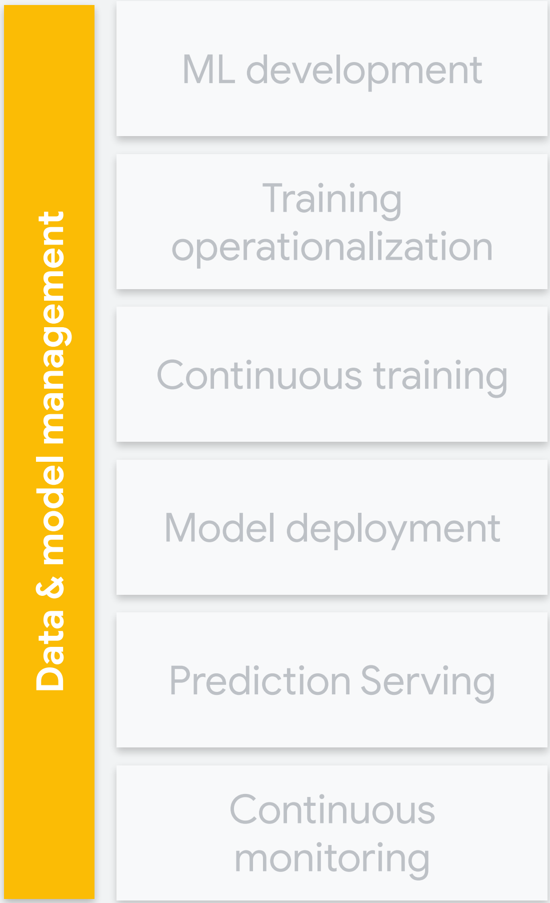
Continuous Monitoring

Continuous monitoring refers to keeping track of a deployed model's effectiveness and efficiency.

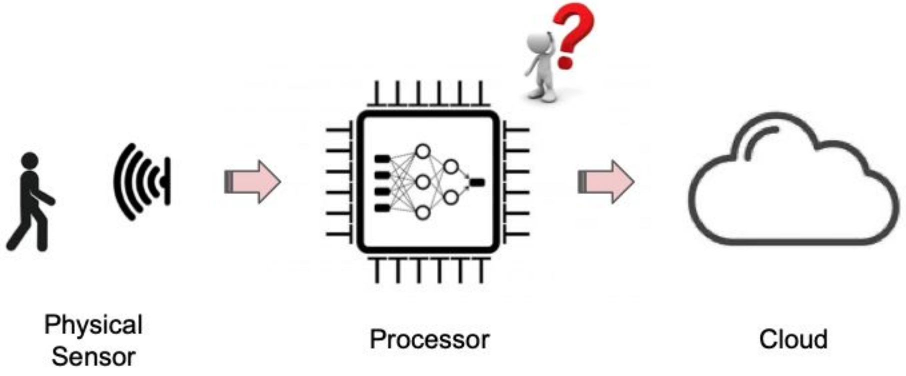


Data & Model Management

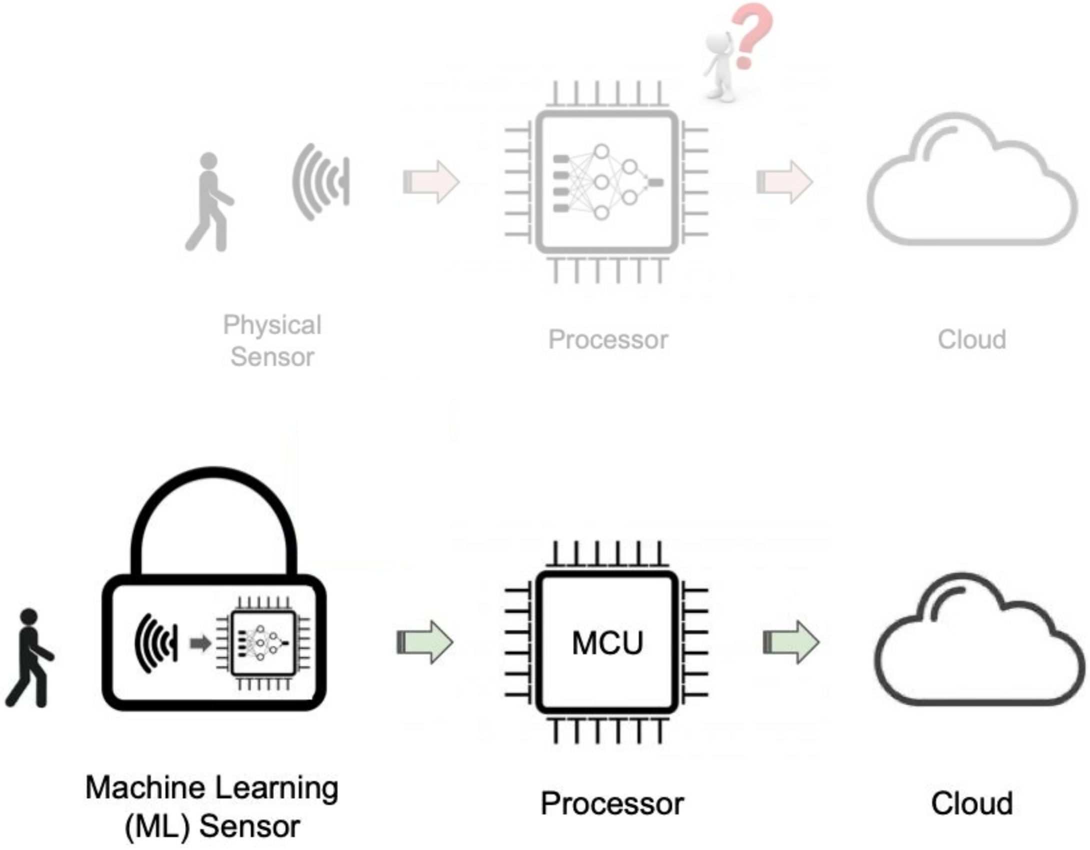
Data and model management is a central, cross-cutting function for governing ML artifacts to support ability, traceability, and compliance. Data and model management can also promote shareability, reusability, and discoverability of ML assets.



ML Sensors



Sensor 1.0



Physical Sensor

Processor

Cloud

Sensor 1.0

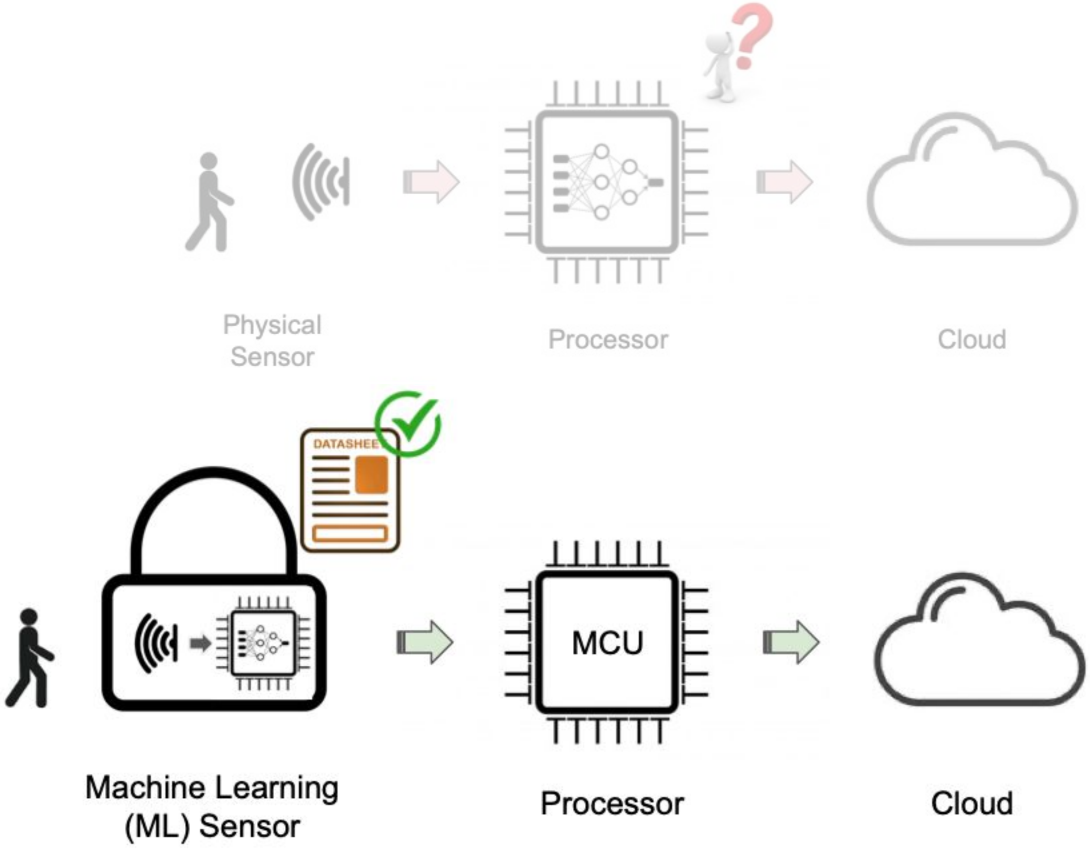
Machine Learning (ML) Sensor

MCU

Processor

Cloud

Sensor 2.0

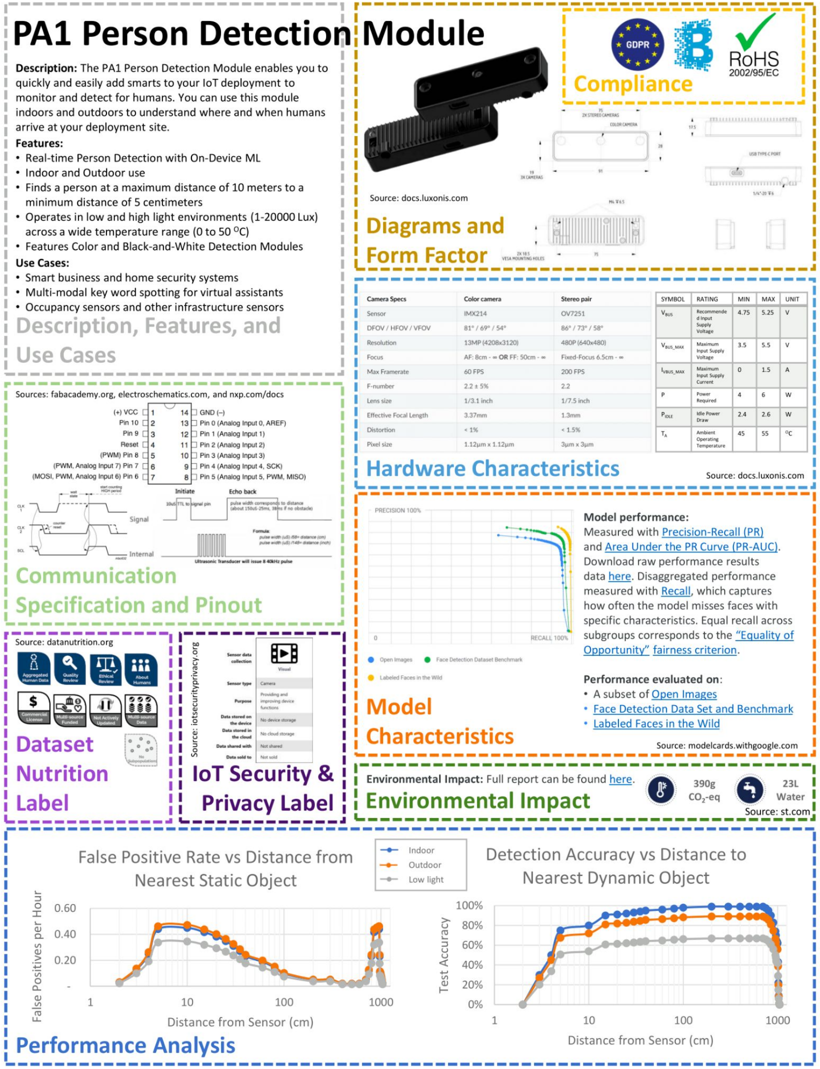


Sensor 1.0

Sensor 2.0

Datasheets for ML Sensors

ML sensors must be **transparent**, indicating in a **publicly and freely accessible** ML sensor **datasheet** all the relevant information such as **fact sheets**, **model cards**, and **dataset nutrition labels** to supplement the traditional EE hardware information typically available for sensors.



PA1 Person Detection Module

Description: The PA1 Person Detection Module enables you to quickly and easily add smarts to your IoT deployment to monitor and detect for humans. You can use this module indoors and outdoors to understand where and when humans arrive at your deployment site.

Features:

- Real-time Person Detection with On-Device ML
- Indoor and Outdoor use
- Finds a person at a maximum distance of 10 meters to a minimum distance of 5 centimeters
- Operates in low and high light environments (1-20000 Lux) across a wide temperature range (0 to 50 ^{O}C)
- Features Color and Black-and-White Detection Modules

Use Cases:

- Smart business and home security systems
- Multi-modal key word spotting for virtual assistants
- Occupancy sensors and other infrastructure sensors

Description, Features, and Use Cases

GDPR RoHS 2002/95/EC

Compliance

Source: docs.luxonis.com

Diagrams and Form Factor

Camera Specs	Color camera	Stereo pair
Sensor	IMX214	OV7251
DFOV / HFOV / VFOV	81° / 69° / 54°	86° / 73° / 58°
Resolution	13MP (4208x3120)	480P (640x480)
Focus	AF: 8cm - ∞ **OR** FF: 50cm - ∞	Fixed-Focus 6.5cm - ∞
Max Framerate	60 FPS	200 FPS
F-number	2.2 ± 5%	2.2
Lens size	1/3.1 inch	1/7.5 inch
Effective Focal Length	3.37mm	1.3mm
Distortion	< 1%	< 1.5%
Pixel size	1.12μm x 1.12μm	3μm x 3μm

SYMBOL	RATING	MIN	MAX	UNIT
V_{BUS}	Recommended Input Supply Voltage	4.75	5.25	V
V_{BUS_MAX}	Maximum Input Supply Voltage	3.5	5.5	V
I_{VBUS_MAX}	Maximum Input Supply Current	0	1.5	A
P	Power Required	4	6	W
P_{IDLE}	Idle Power Draw	2.4	2.6	W
T_A	Ambient Operating Temperature	45	55	^{O}C

Hardware Characteristics

Source: docs.luxonis.com

Sources: fabacademy.org, electroschematics.com, and nxp.com/docs

(+) VCC 1
Pin 10 2
Pin 9 3
Reset 4
(PWM) Pin 8 5
(PWM, Analog Input 7) Pin 7 6
(MOSI, PWM, Analog Input 6) Pin 6 7
14 GND (–)
13 Pin 0 (Analog Input 0, AREF)
12 Pin 1 (Analog Input 1)
11 Pin 2 (Analog Input 2)
10 Pin 3 (Analog Input 3)
9 Pin 4 (Analog Input 4, SCK)
8 Pin 5 (Analog Input 5, PWM, MISO)
wait state
start counting HIGH period
CLK 1
CLK 2
counter reset
SCL
Signal
Internal
Initiate
Echo back
10uS TTL to signal pin
pulse width corresponds to distance (about 150uS-25ms, 38ms if no obstacle)
Formula:
pulse width (uS) /58= distance (cm)
pulse width (uS) /148= distance (inch)
Ultrasonic Transducer will issue 8 40kHz pulse

Communication Specification and Pinout

Model performance:
Measured with Precision-Recall (PR) and Area Under the PR Curve (PR-AUC). Download raw performance results data here. Disaggregated performance measured with Recall, which captures how often the model misses faces with specific characteristics. Equal recall across subgroups corresponds to the "Equality of Opportunity" fairness criterion.

Performance evaluated on:

- A subset of Open Images
- Face Detection Data Set and Benchmark
- Labeled Faces in the Wild

Model Characteristics

Source: modelcards.withgoogle.com

Source: datanutrition.org

Dataset Nutrition Label

Source: iotsecurityprivacy.org

Sensor data collection	Visual
Sensor type	Camera
Purpose	Providing and improving device functions
Data stored on the device	No device storage
Data stored in the cloud	No cloud storage
Data shared with	Not shared
Data sold to	Not sold

IoT Security & Privacy Label

Environmental Impact: Full report can be found here.

390g CO_2-eq

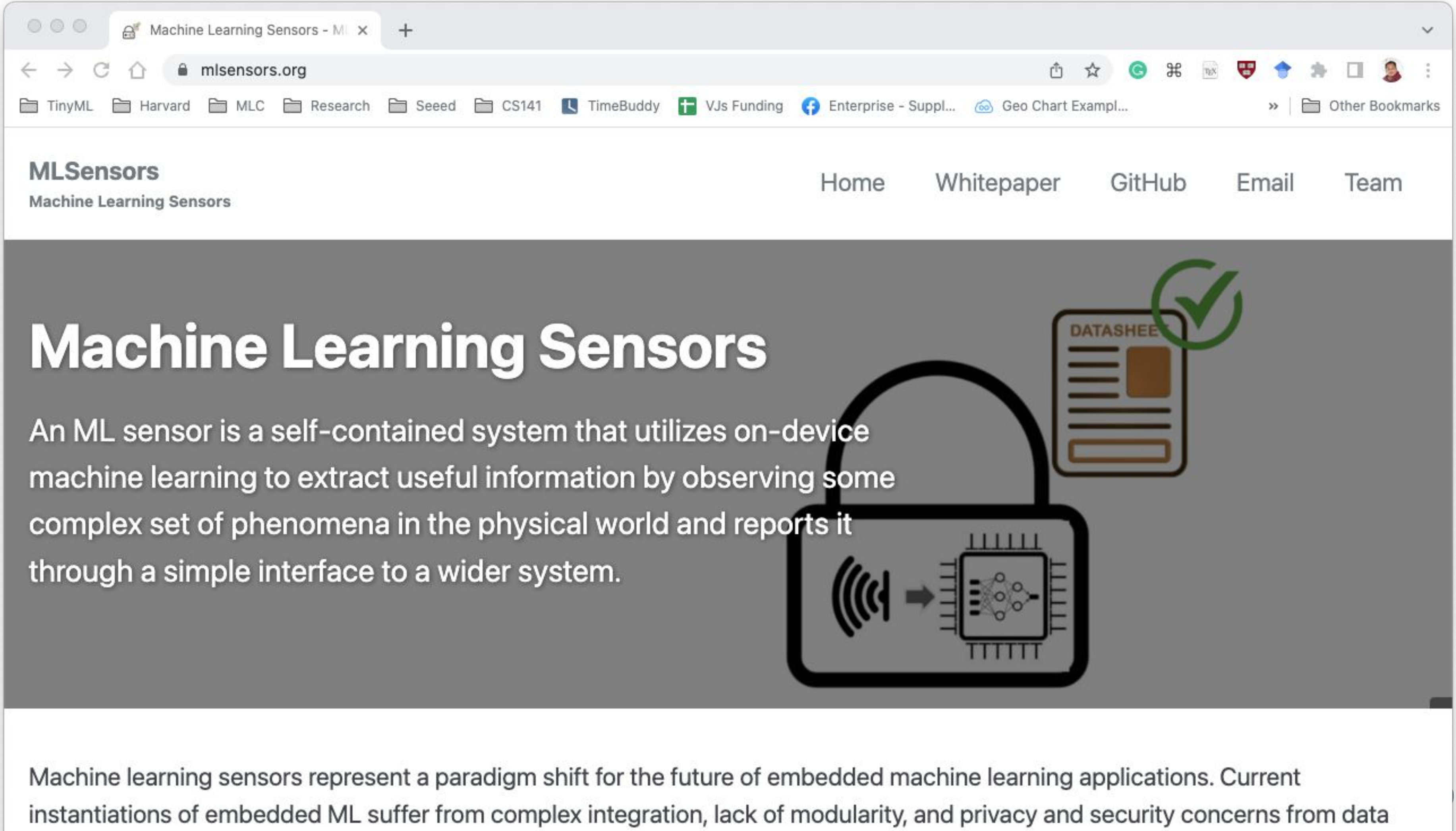
23L Water

Environmental Impact

Source: st.com

False Positive Rate vs Distance from Nearest Static Object
Indoor
Outdoor
Low light
False Positives per Hour
0.60
0.40
0.20
-
1
10
100
1000
Distance from Sensor (cm)
Detection Accuracy vs Distance to Nearest Dynamic Object
Test Accuracy
100%
80%
60%
40%
20%
0%
1
10
100
1000
Distance from Sensor (cm)

Performance Analysis



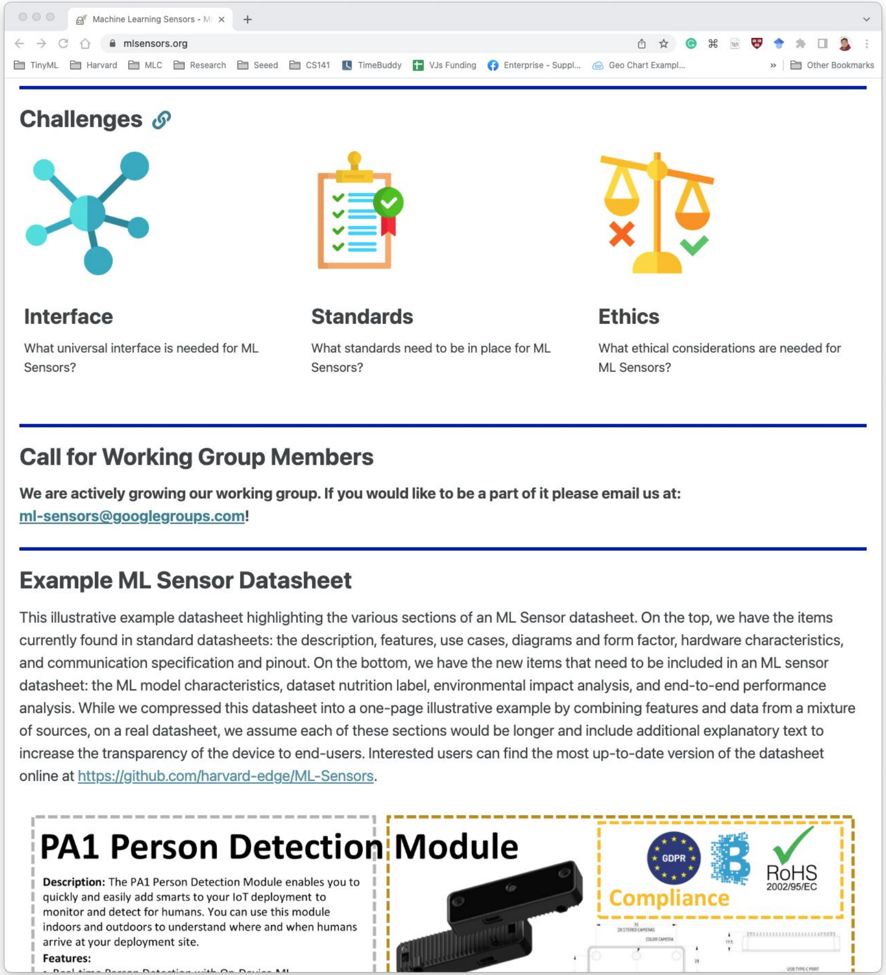
MLSensors

Machine Learning Sensors

Machine Learning Sensors

An ML sensor is a self-contained system that utilizes on-device machine learning to extract useful information by observing some complex set of phenomena in the physical world and reports it through a simple interface to a wider system.

Machine learning sensors represent a paradigm shift for the future of embedded machine learning applications. Current instantiations of embedded ML suffer from complex integration, lack of modularity, and privacy and security concerns from data



Challenges

Interface

What universal interface is needed for ML Sensors?

Standards

What standards need to be in place for ML Sensors?

Ethics

What ethical considerations are needed for ML Sensors?

Call for Working Group Members

We are actively growing our working group. If you would like to be a part of it please email us at: ml-sensors@googlegroups.com!

Example ML Sensor Datasheet

This illustrative example datasheet highlighting the various sections of an ML Sensor datasheet. On the top, we have the items currently found in standard datasheets: the description, features, use cases, diagrams and form factor, hardware characteristics, and communication specification and pinout. On the bottom, we have the new items that need to be included in an ML sensor datasheet: the ML model characteristics, dataset nutrition label, environmental impact analysis, and end-to-end performance analysis. While we compressed this datasheet into a one-page illustrative example by combining features and data from a mixture of sources, on a real datasheet, we assume each of these sections would be longer and include additional explanatory text to increase the transparency of the device to end-users. Interested users can find the most up-to-date version of the datasheet online at https://github.com/harvard-edge/ML-Sensors.

PA1 Person Detection Module

Description: The PA1 Person Detection Module enables you to quickly and easily add smarts to your IoT deployment to monitor and detect for humans. You can use this module indoors and outdoors to understand where and when humans arrive at your deployment site.

Features:

ML Sensors - Guiding Set of Principles

1. We need to **raise the level of abstraction** to enable ease of use for scalable deployment of ML sensors; not everyone should be required to be an systems developer or an engineer to use or leverage ML sensors into their ecosystem.

2. The ML sensor's **design should be inherently data-centric** and defined by its input-output behavior instead of exposing the underlying hardware and software mechanisms that support ML model execution.

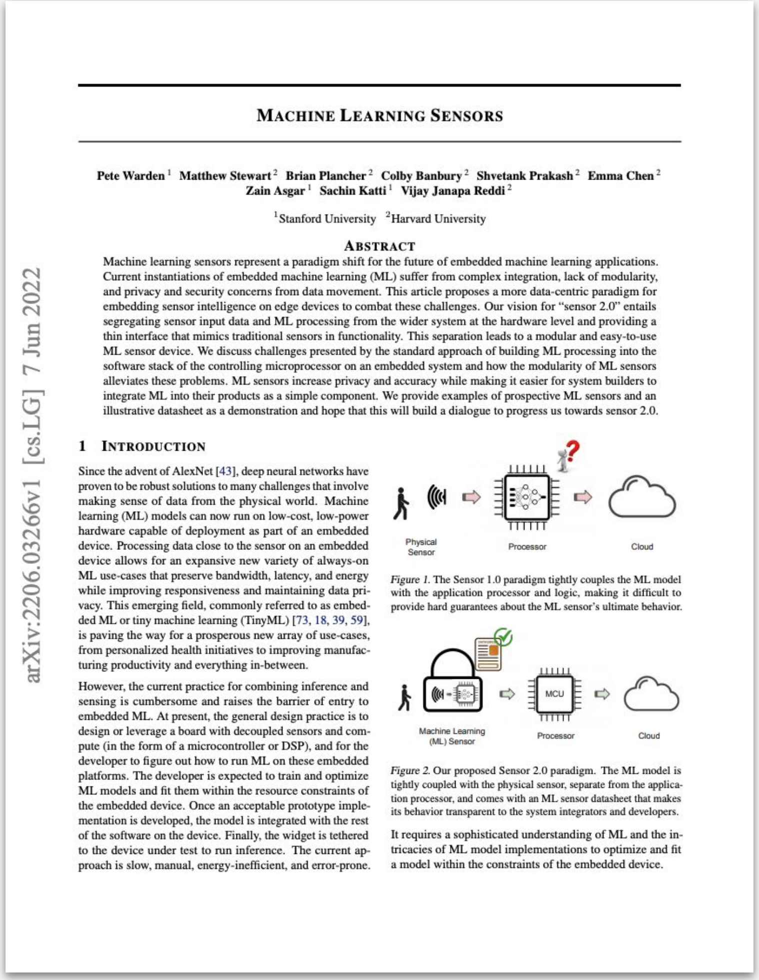
3. An ML sensor's **implementation must be clean and complexity-free**. Features such as reusability, software updates, and networking must be thought through to ensure data privacy and secure execution.

4. ML sensors **must be transparent, indicating in a publicly and freely accessible ML sensor datasheet** all the relevant information such as fact sheets, model cards, and dataset nutrition labels to supplement the traditional information available for hardware sensors.

5. We as a community should aim to **foster an open ML sensors ecosystem by maximizing data, model, and hardware transparency** where possible, without necessarily relinquishing any claim to intellectual property.

Machine Learning Sensors

“An ML sensor is self-contained system that utilizes on-device machine learning extract useful information by observing some complex set of phenomena in the physical world and reports it through a simple interface to a wider system.”



Machine Learning Sensors

Pete Warden[1] Matthew Stewart[2] Brian Plancher[2] Colby Banbury[2] Shvetank Prakash[2] Emma Chen[2]
Zain Asgar[1] Sachin Katti[1] Vijay Janapa Reddi[2]

[1]Stanford University [2]Harvard University

arXiv:2206.03266v1 [cs.LG] 7 Jun 2022

Abstract

Machine learning sensors represent a paradigm shift for the future of embedded machine learning applications. Current instantiations of embedded machine learning (ML) suffer from complex integration, lack of modularity, and privacy and security concerns from data movement. This article proposes a more data-centric paradigm for embedding sensor intelligence on edge devices to combat these challenges. Our vision for “sensor 2.0” entails segregating sensor input data and ML processing from the wider system at the hardware level and providing a thin interface that mimics traditional sensors in functionality. This separation leads to a modular and easy-to-use ML sensor device. We discuss challenges presented by the standard approach of building ML processing into the software stack of the controlling microprocessor on an embedded system and how the modularity of ML sensors alleviates these problems. ML sensors increase privacy and accuracy while making it easier for system builders to integrate ML into their products as a simple component. We provide examples of prospective ML sensors and an illustrative datasheet as a demonstration and hope that this will build a dialogue to progress us towards sensor 2.0.

1 Introduction

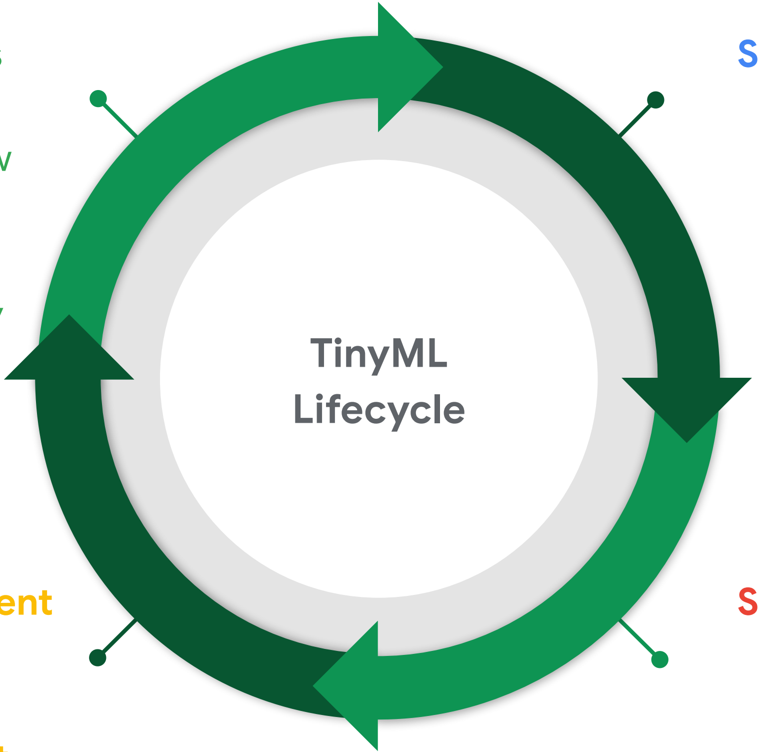
Since the advent of AlexNet [43], deep neural networks have proven to be robust solutions to many challenges that involve making sense of data from the physical world. Machine learning (ML) models can now run on low-cost, low-power hardware capable of deployment as part of an embedded device. Processing data close to the sensor on an embedded device allows for an expansive new variety of always-on ML use-cases that preserve bandwidth, latency, and energy while improving responsiveness and maintaining data privacy. This emerging field, commonly referred to as embedded ML or tiny machine learning (TinyML) [73, 18, 39, 59], is paving the way for a prosperous new array of use-cases, from personalized health initiatives to improving manufacturing productivity and everything in-between.

However, the current practice for combining inference and sensing is cumbersome and raises the barrier of entry to embedded ML. At present, the general design practice is to design or leverage a board with decoupled sensors and compute (in the form of a microcontroller or DSP), and for the developer to figure out how to run ML on these embedded platforms. The developer is expected to train and optimize ML models and fit them within the resource constraints of the embedded device. Once an acceptable prototype implementation is developed, the model is integrated with the rest of the software on the device. Finally, the widget is tethered to the device under test to run inference. The current approach is slow, manual, energy-inefficient, and error-prone.

Figure 1. The Sensor 1.0 paradigm tightly couples the ML model with the application processor and logic, making it difficult to provide hard guarantees about the ML sensor’s ultimate behavior.

Figure 2. Our proposed Sensor 2.0 paradigm. The ML model is tightly coupled with the physical sensor, separate from the application processor, and comes with an ML sensor datasheet that makes its behavior transparent to the system integrators and developers.

It requires a sophisticated understanding of ML and the intricacies of ML model implementations to optimize and fit a model within the constraints of the embedded device.



Stage 1: Sensor Data

- Heterogeneous devices
- Multi-modal data
- Sensor drift
- Varying data frequency

Stage 2: Model Development

- ML model architecture
- Resource constraints
- Model quality/accuracy
- End-to-end performance

Stage 3: Model Deployment

- Model robustness
- Scalable deployment
- Embedded MLOps
- System Integration

Stage 4: Live Operations

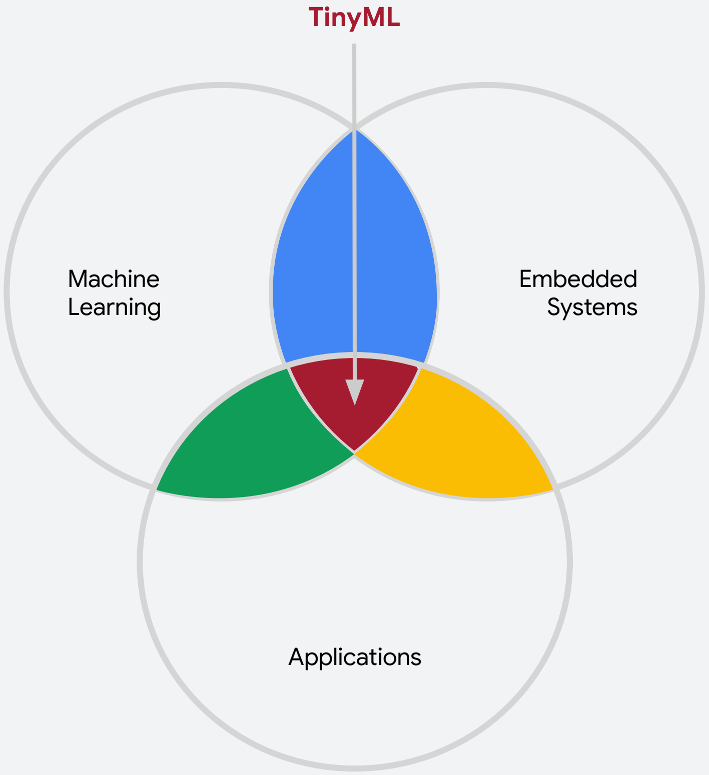
- Model drift and skew
- Ethical challenges
- Sustainability
- Security and privacy

Course Topics

1. **Overview and Introduction to Embedded Machine Learning**
2. Data Engineering
3. Embedded Machine Learning Frameworks
4. Efficient Model Representation and Compression
5. Performance Metrics and Benchmarking of ML Systems
6. Learning on the Edge
7. Hardware Acceleration for Edge ML: GPUs, TPUs and FPGAs
8. Embedded MLOps
9. Secure and Privacy-Preserving On-Device ML
10. Responsible AI
11. Sustainability at the Edge
12. Generative AI at the Edge

Conclusion

1. TinyML has the **potential to radically change our future**
2. No free lunch – hardware and software **fragmentation is a serious challenge** to address
3. TinyML **sustainability is crucial** to ensure its broad applicability
4. ML sensors based on TinyML technology must be **transparent**
5. Widening access to applied ML is a must to ensure **equitable access**



*The future of ML is tiny and bright,
and its benefits can translate to societal impact.*

CS249r Fall 2022 Projects

Unilateral Gait Event Classification by Measuring Local Muscle Deformation

Jonathan Alvarez

Thanks also to Dabin Choe, Ariane Daney De Marcillac, James Arnold, and Colby Banbury

CS249R: Tiny ML

12.2022

Discussion Topics

Understanding TinyML

What is tinyML and how does it differ from traditional machine learning approaches?

What are the potential applications of tinyML in everyday life?

How does tinyML align with the current trends in the Internet of Things (IoT)?

Class Discussion

Application and Use Cases

What kind of real-world problems can be solved more efficiently with tinyML compared to traditional ML solutions?

How can tinyML contribute to energy conservation and sustainability?

What industries or sectors could benefit the most from tinyML technologies?

Security and Privacy

What are the potential security vulnerabilities associated with deploying tinyML in sensitive applications?
How can privacy be maintained when deploying tinyML solutions in personal devices?
How can tinyML aid in the development of secure communication networks?

Ethics and Society

What are the potential societal impacts of widespread adoption of tinyML technologies?

What ethical considerations should be kept in mind when deploying tinyML in public spaces?

How can tinyML technologies be made accessible and inclusive for different communities?