TinyML Challenges

TINY ML

Hardware

Software

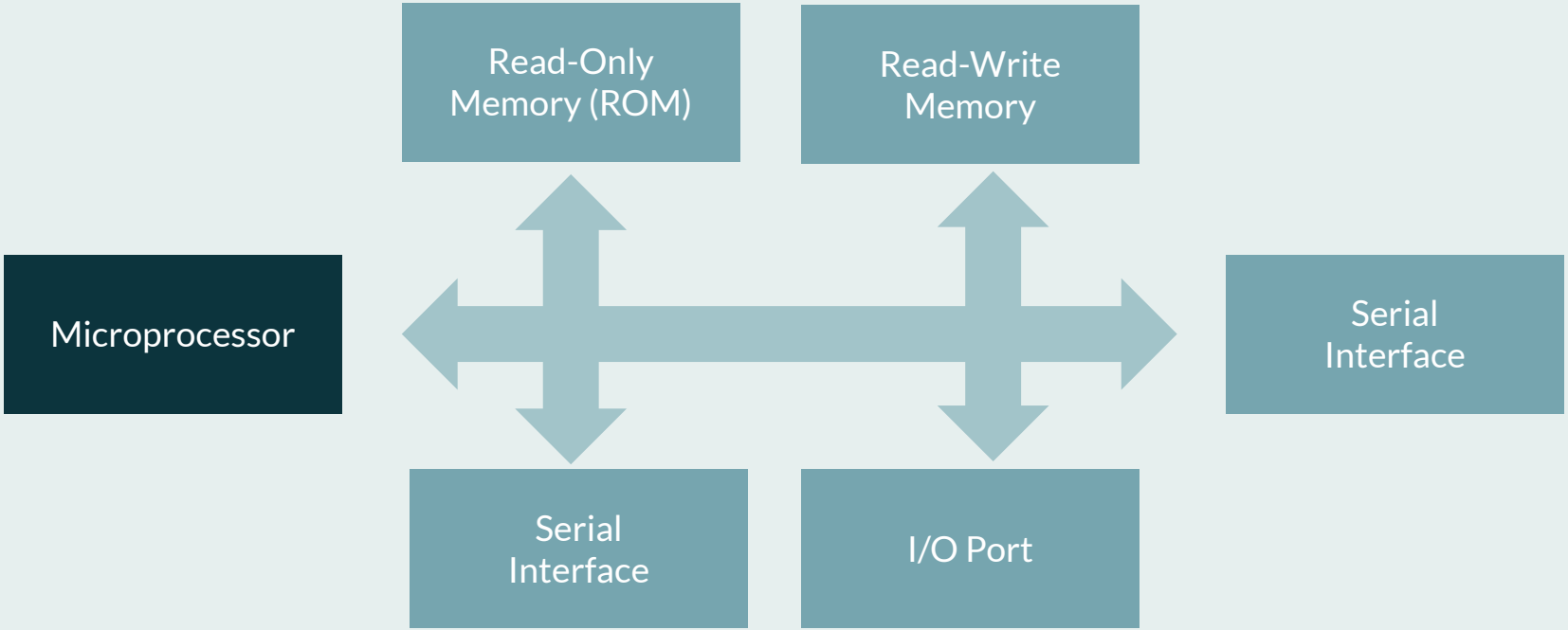
Compute

Memory

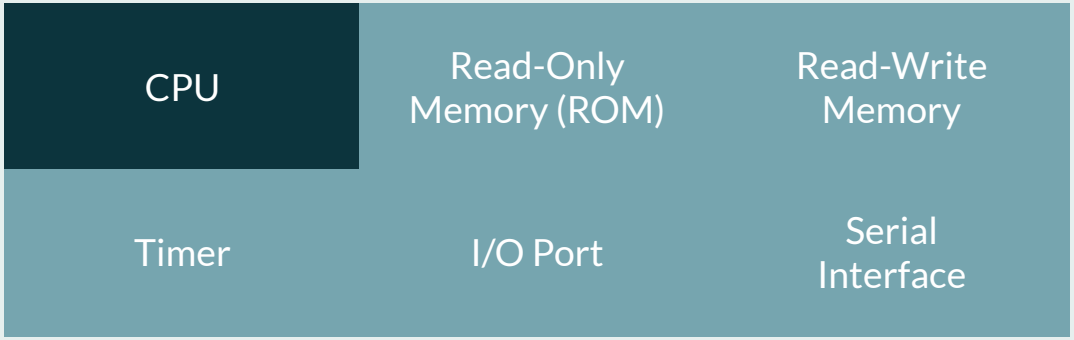
Storage

Microprocessor Vs Microcontroller

Microprocessor: only **one part** of the puzzle



Microcontroller



Microprocessor

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

Microcontroller

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

Orders of Magnitude Difference

	Microprocessor	>	Microcontroller
Platform			
Compute	1GHz - 4GHz	10x	1MHz - 400MHz
Memory	512MB - 64GB	10Kx	2KB - 512KB
Storage	64GB - 4TB	100Kx	32KB - 2MB
Power	30W - 100W	1Kx	150μW - 23.5mW

Microcontroller

1MHz - 400MHz

2KB - 512KB

32KB - 2MB

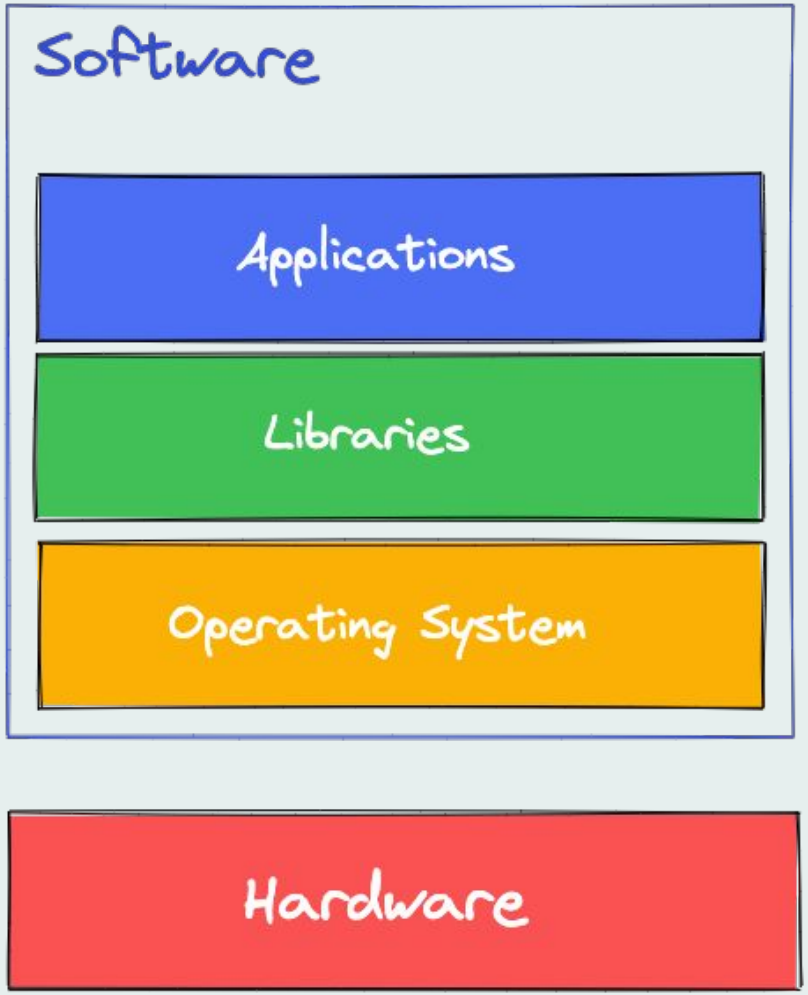
150μW - 23.5mW

Implications

- How complicated is the running task?
- How much memory does it need to have?
- How long does the job have to perform?

Hardware

Software

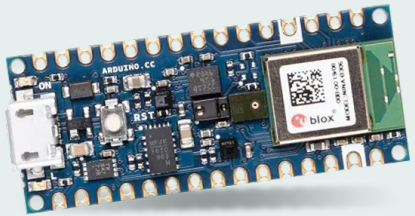

Software
Applications
Libraries
Operating System
Hardware

Widely Used Operating Systems

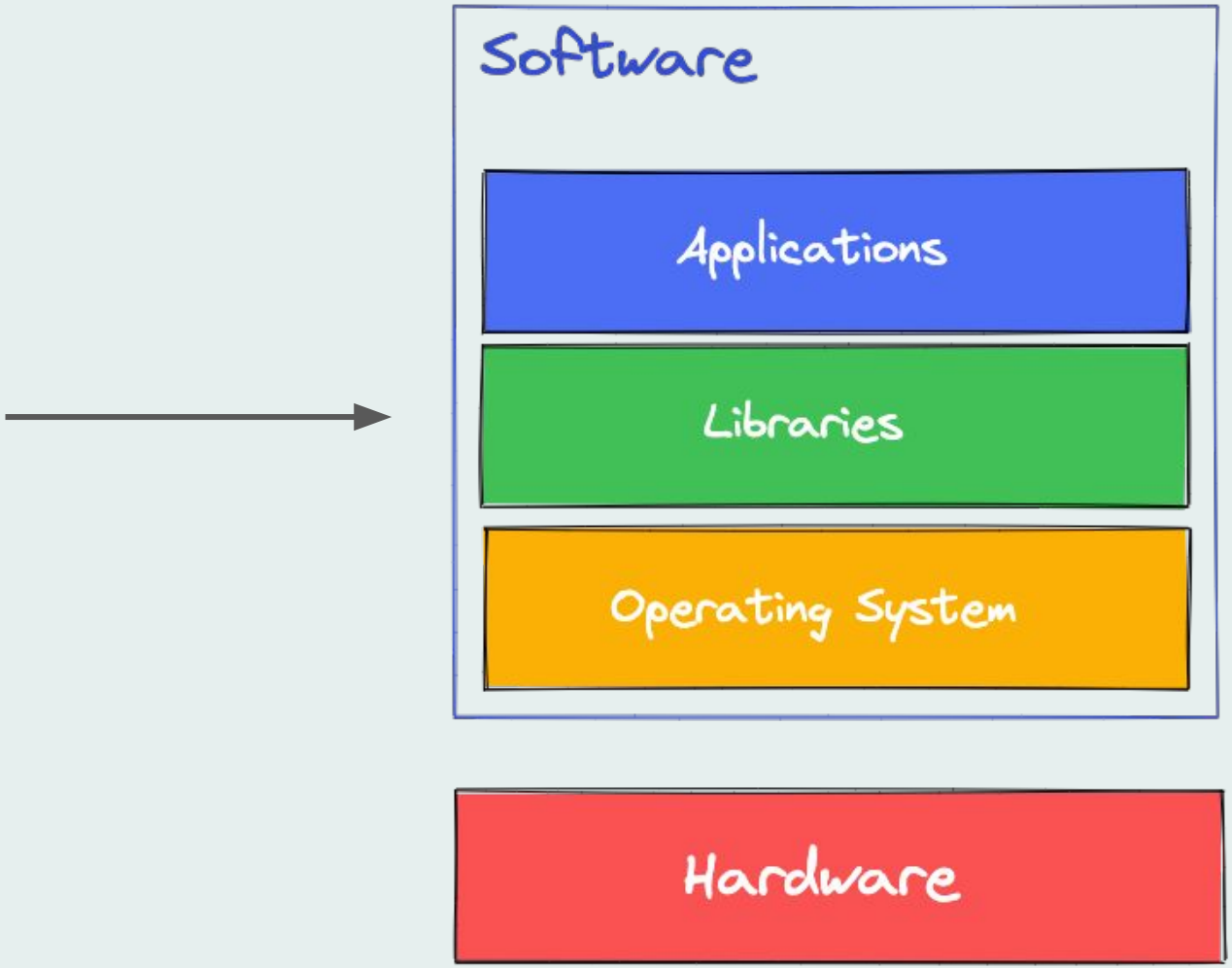
The Android Software Stack

Widely Used Operating Systems

Embedded Systems



Widely Used Operating Systems


Software
Applications
Libraries
Operating System
Hardware

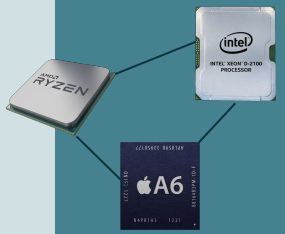
Software

Applications

Libraries

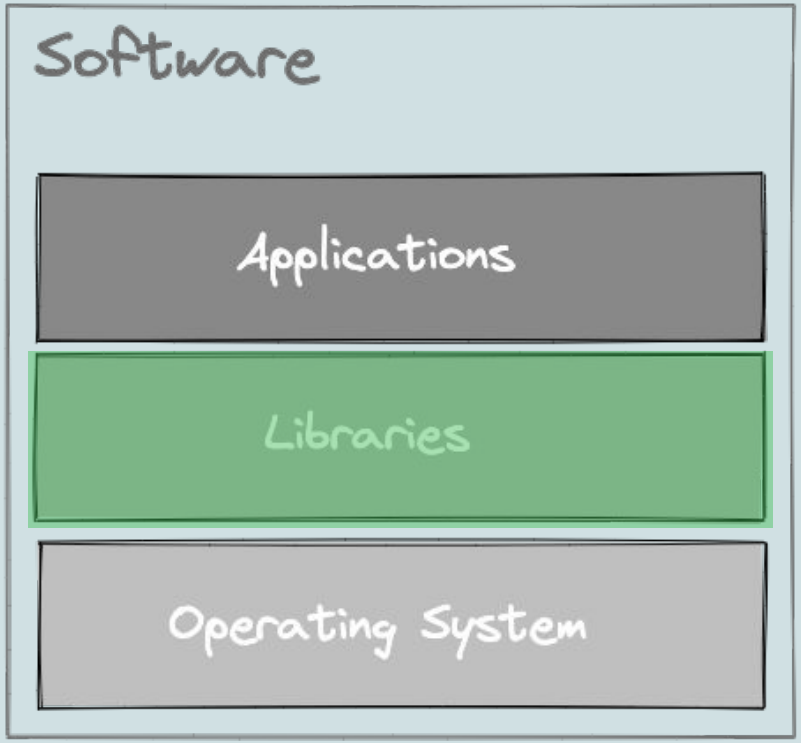
Operating System

Hardware



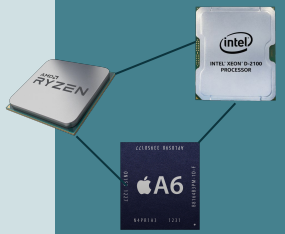

```
import numpy as np

For x in range(10):
    np.SaveTheWorld()
```



Portability Opportunity

Able to execute the same code on different microprocessor hardware and architecture

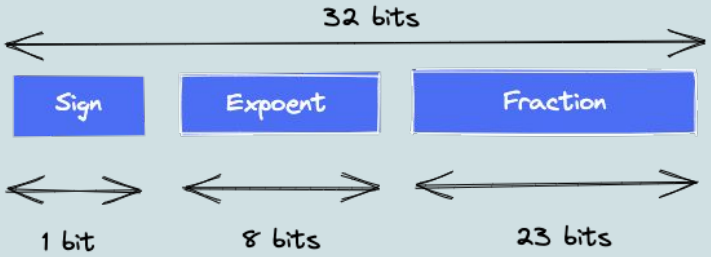


π

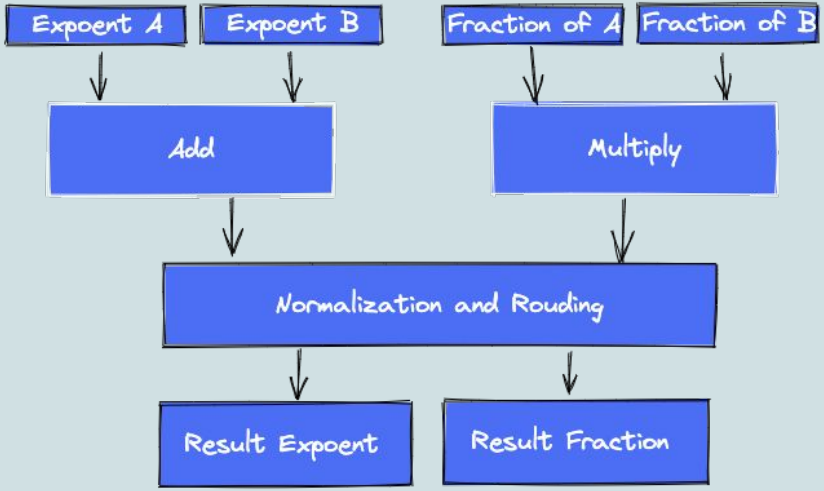
$\frac{22}{7}$

π

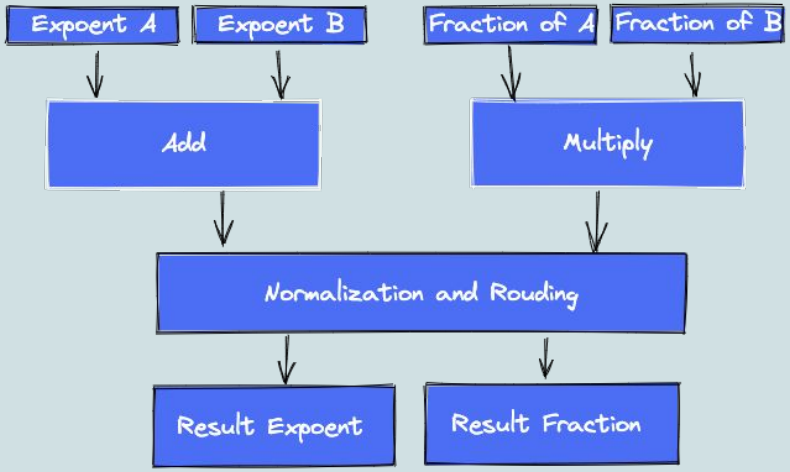
3.1415926535897932384626433832795028841971⋱
6939937510582097494459230781640628620899⋱
8628034825342117067982148086513282306647⋱
0938446095505822317253594081284811174502⋱
8410270193852110555964462294895493038196⋱
4428810975665933446128475648233786783165⋱
2712019091456485669234603486104543266482⋱
1339360726024914127372458700660631558817⋱
4881520920962829254091715364367892590360⋱
0113305305488204665213841469519415116094⋱
3305727036575959195309218611738193261179⋱
3105118548074462379962749567351885752724⋱
8912279381830119491298336733624406566430⋱
8602139494639522473719070217986094370277⋱
0539217176293176752384674818467669405132⋱
0005681271452635608277857713427577896091⋱
7363717872146844090122495343014654958537⋱
1050792279689258923542019956112129021960⋱
8640344181598136297747713099605187072113⋱
4999999837297804995105973173281609631859⋱
5024459455346908302642522308253344685035⋱
2619311881710100031378387528865875332083⋱
8142061717766914730359825349042875546873⋱
1159562863882353787593751957781857780532⋱
171226806613001927876611195909216420199



Single Precision
IEEE 754 Floating-Point Standard

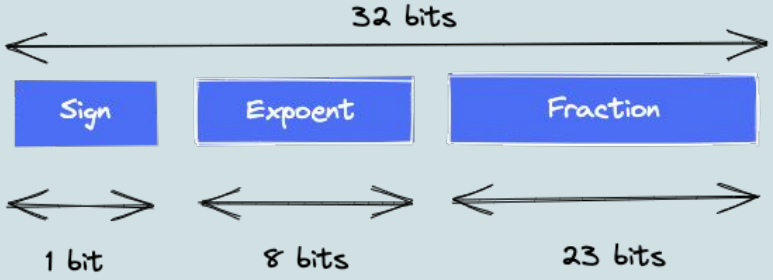


3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036011330530548820466521384146951941511609433057270365759591953092186117381932611793105118548074462379962749567351885752724891227938183011949129833673362440656643086021394946395224737190702179860943702770539217176293176752384674818467669405132000568127145263560827785771342757789609173637178721468440901224953430146549585371050792279689258923542019956112129021960864034418159813629774771309960518707211349999983729780499510597317328160963185950244594553469083026425223082533446850352619311881710100031378387528865875332083814206171776691473035982534904287554687311595628638823537875937519577818577805321712268066130019278766119590921642019



Libraries

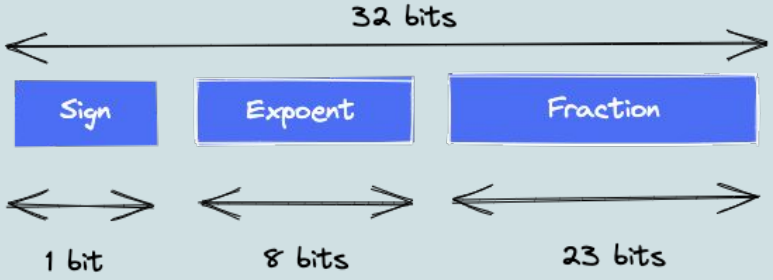
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036011330530548820466521384146951941511609433057270365759591953092186117381932611793105118548074462379962749567351885752724891227938183011949129833673362440656643086021394946395224737190702179860943702770539217176293176752384674818467669405132000568127145263560827785771342757789609173637178721468440901224953430146549585371050792279689258923542019956112129021960864034418159813629774771309960518707211349999983729780499510597317328160963185950244594553469083026425223082533446850352619311881710100031378387528865875332083814206171776691473035982534904287554687311595628638823537875937519577818577805321712268066130019278766111959092164201995



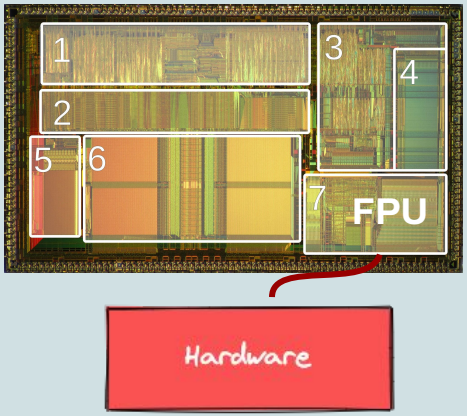
Single Precision
IEEE 754 Floating-Point Standard

Hardware

3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036011330530548820466521384146951941511609433057270365759591953092186117381932611793105118548074462379962749567351885752724891227938183011949129833673362440656643086021394946395224737190702179860943702770539217176293176752384674818467669405132000568127145263560827785771342757789609173637178721468440901224953430146549585371050792279689258923542019956112129021960864034418159813629774771309960518707211349999998372978049951059731732816096318595024459455346908302642522308253344685035261931188171010003137838752886587533208381420617177669147303598253490428755468731159562863882353787593751957781857780532171226806613001927876611195909216420199

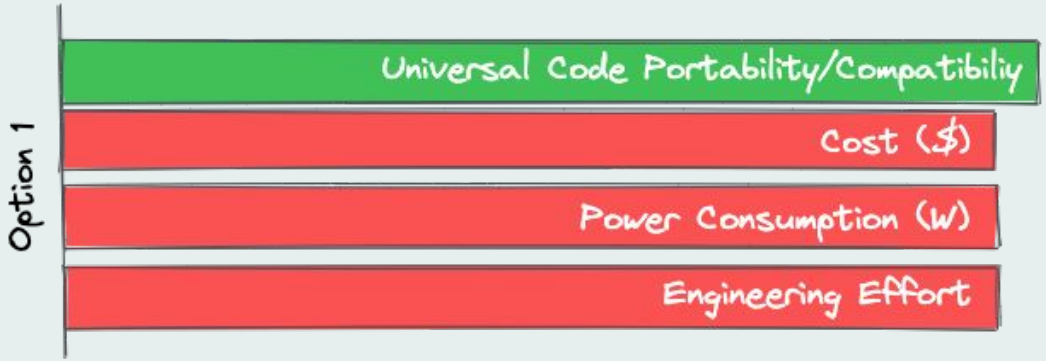


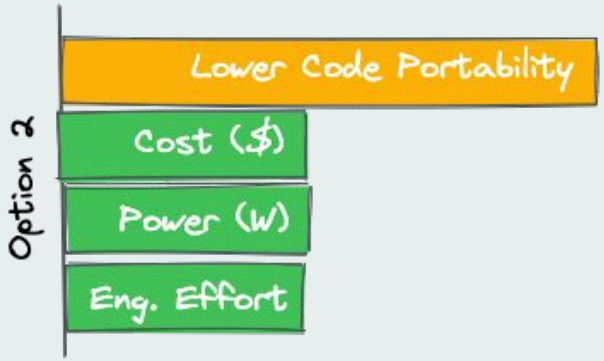
Single Precision
IEEE 754 Floating-Point Standard



3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036011330530548820466521384146951941511609433057270365759591953092186117381932611793105118548074462379962749567351885752724891227938183011949129833673362440656643086021394946395224737190702179860943702770539217176293176752384674818467669405132000568127145263560827785771342757789609173637178721468440901224953430146549585371050792279689258923542019956112129021960864034418159813629774771309960518707211349999998372978049951059731732816096318595024459455346908302642522308253344685035261931188171010003137838752886587533208381420617177669147303598253490428755468731159562863882353787593751957781857780532171226806613001927876611195909216420199

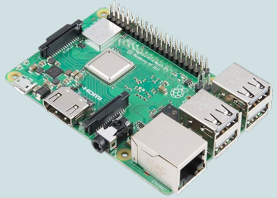
Portability Trade-offs

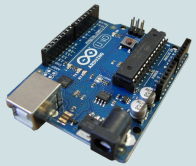




Portability Trade-offs

Sacrifice portability across systems for efficiency in system performance and power efficiency

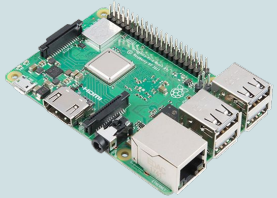


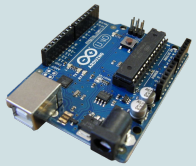


Specific HW Implementation of a Library

Portability Trade-offs

Sacrifice portability across systems for efficiency in system performance and power efficiency





Specific HW Implementation of a Library

Question:

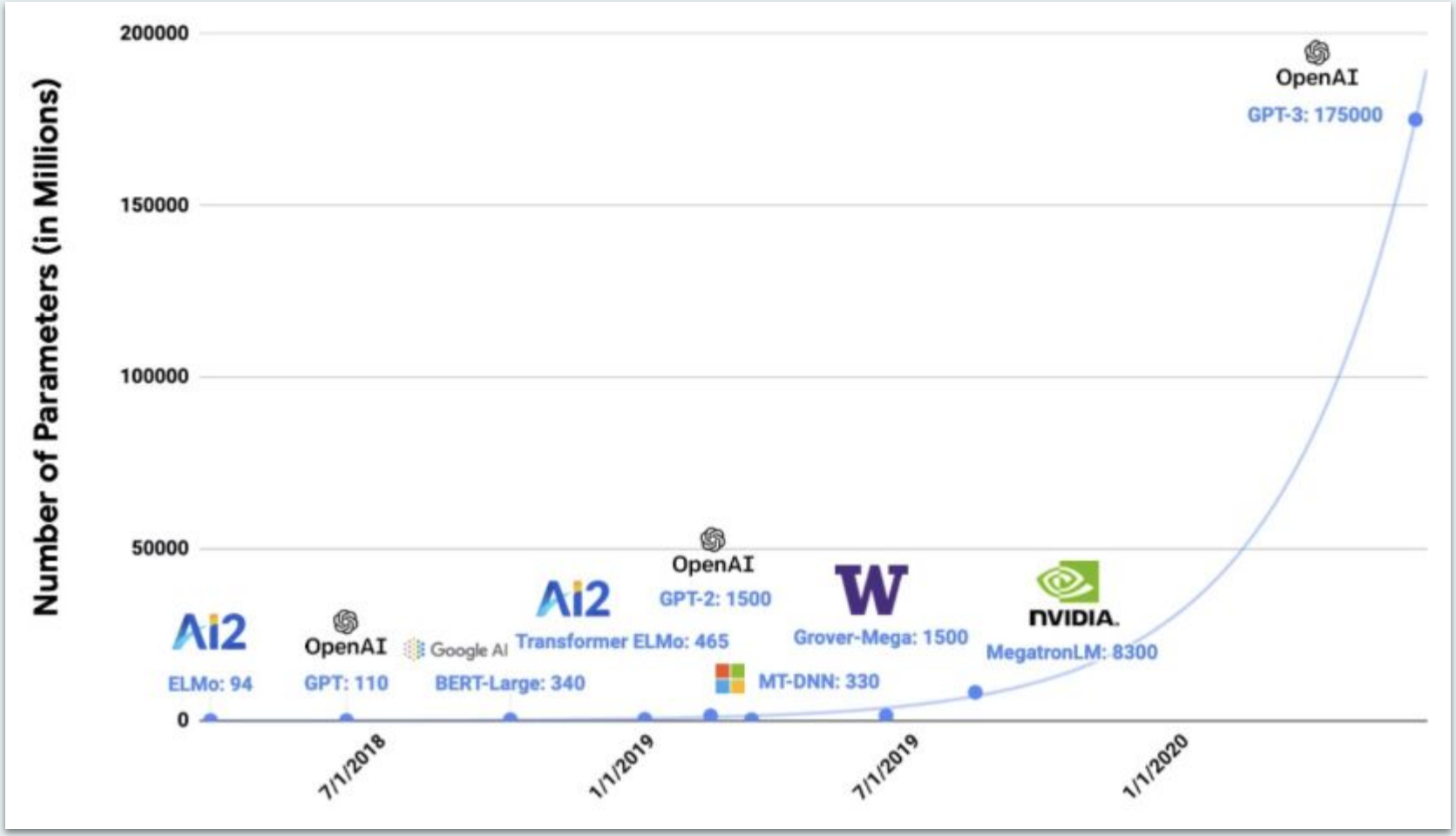
How do we enable TinyML uniformly across these different systems if there is lower platform portability?

Summary

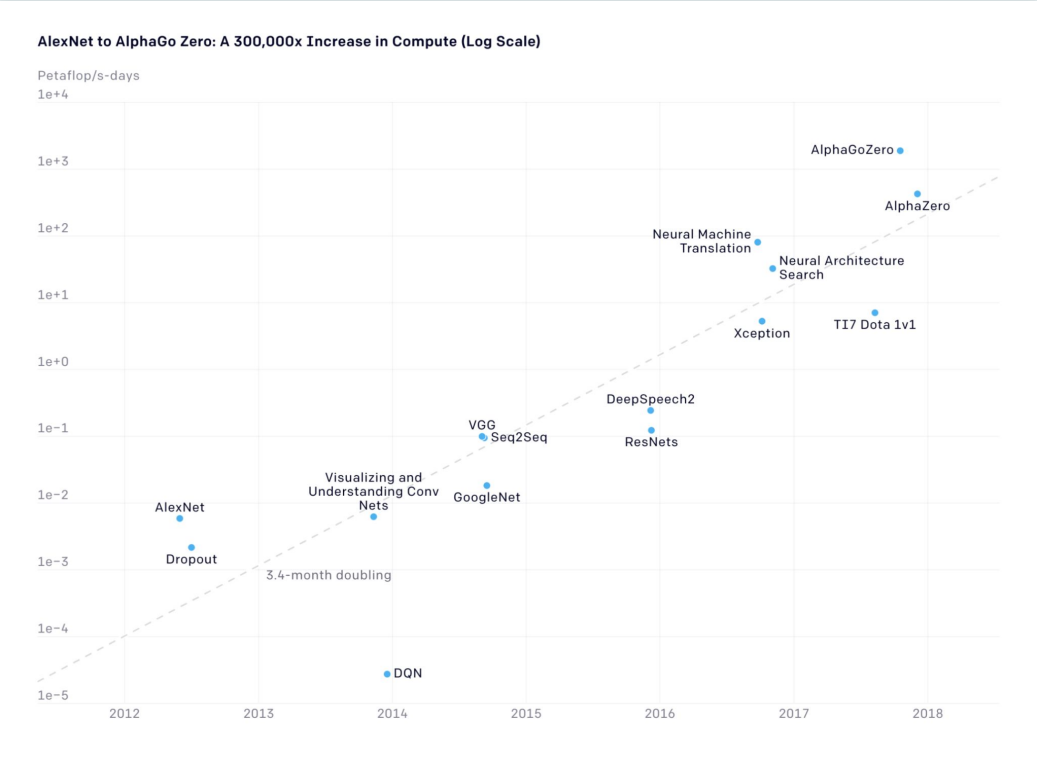
Embedded hardware is extremely limited in performance, power consumption and storage

Embedded software is not as portable and flexible as mainstream computing

TinyML

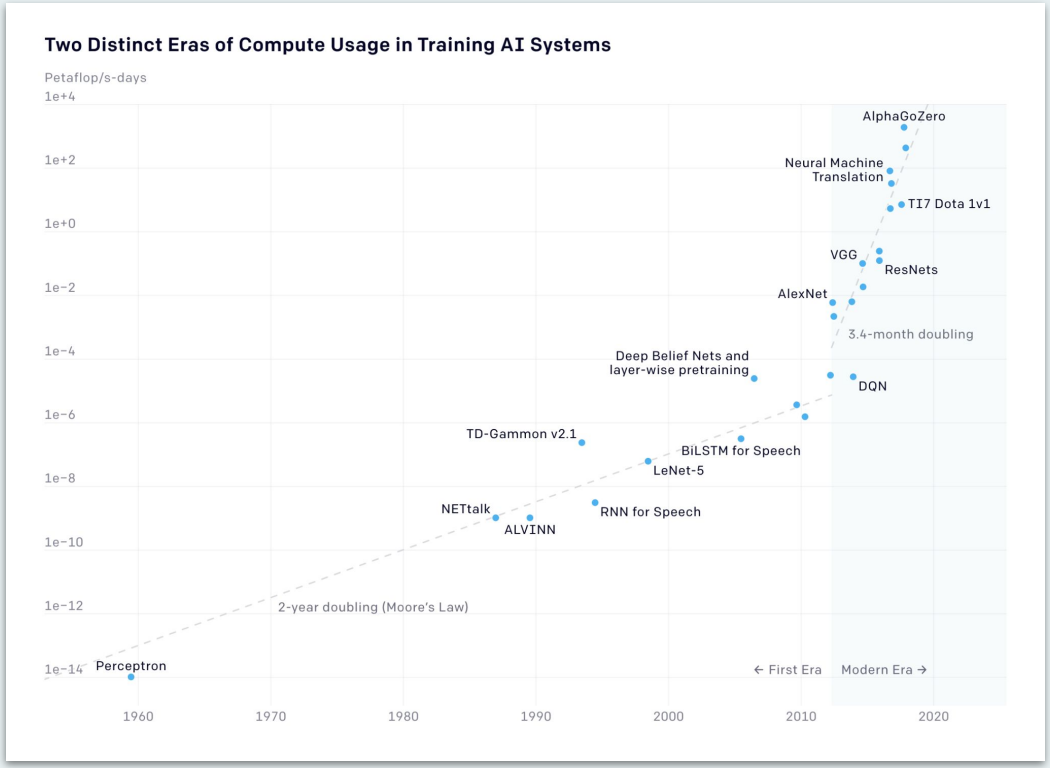

Number of Parameters (in Millions)
200000
150000
100000
50000
0
7/1/2018
1/1/2019
7/1/2019
1/1/2020
Ai2
ELMo: 94
OpenAI
GPT: 110
Google AI
BERT-Large: 340
Ai2
Transformer ELMo: 465
OpenAI
GPT-2: 1500
MT-DNN: 330
W
Grover-Mega: 1500
NVIDIA.
MegatronLM: 8300
OpenAI
GPT-3: 175000

ML Compute Needs (2012 to Present Day)



In recent years, **computing needs grew by 300,000x** to train the machine learning models that are widely deployed in the industry

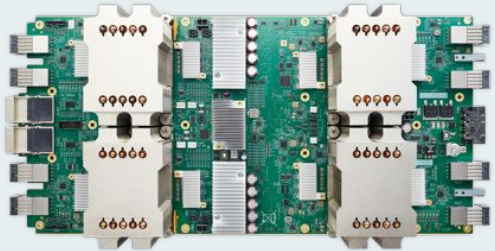
ML Compute Needs (from the 1960s)



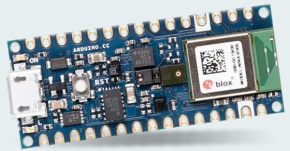
In recent years, the amount of computing needed has grown remarkably fast.

Computer requirements are **doubling nearly every 3 to 4 months.**

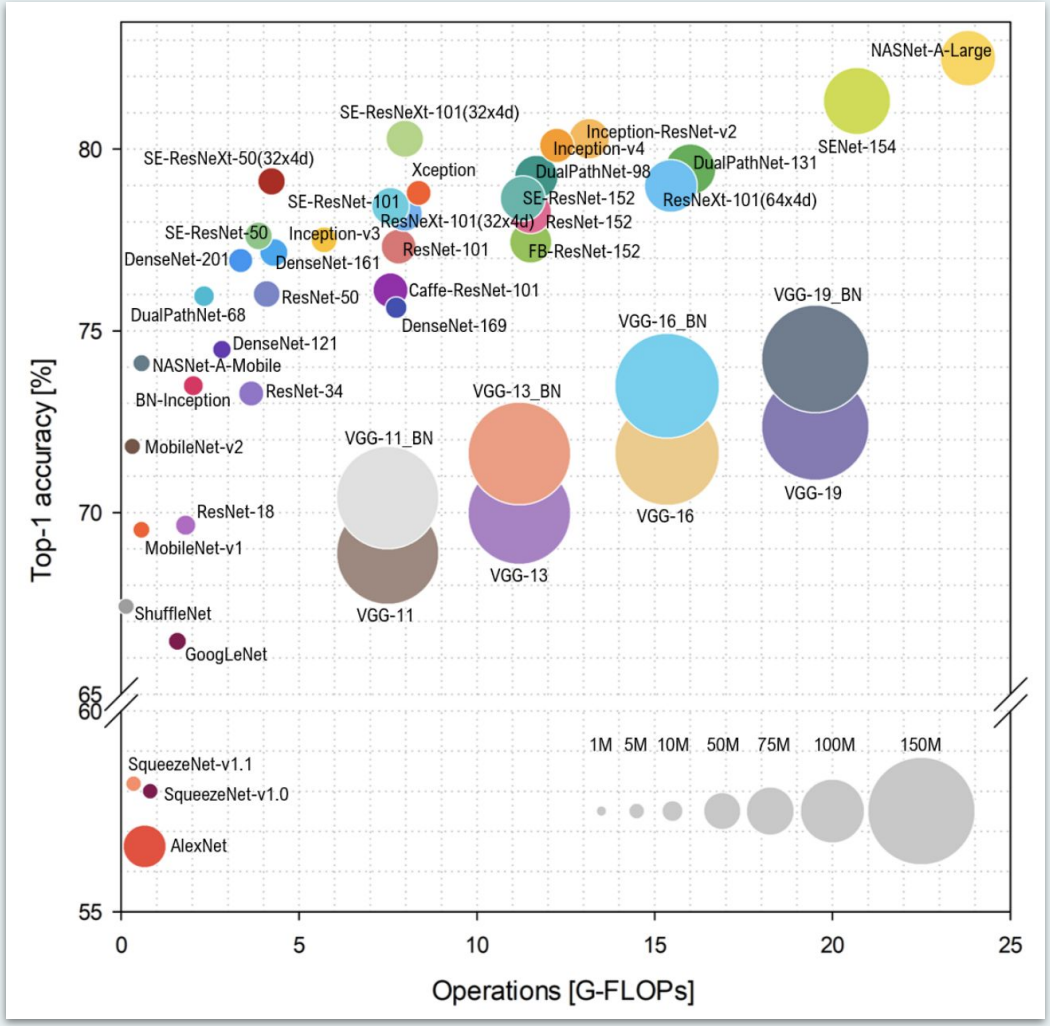
Cloud TPU



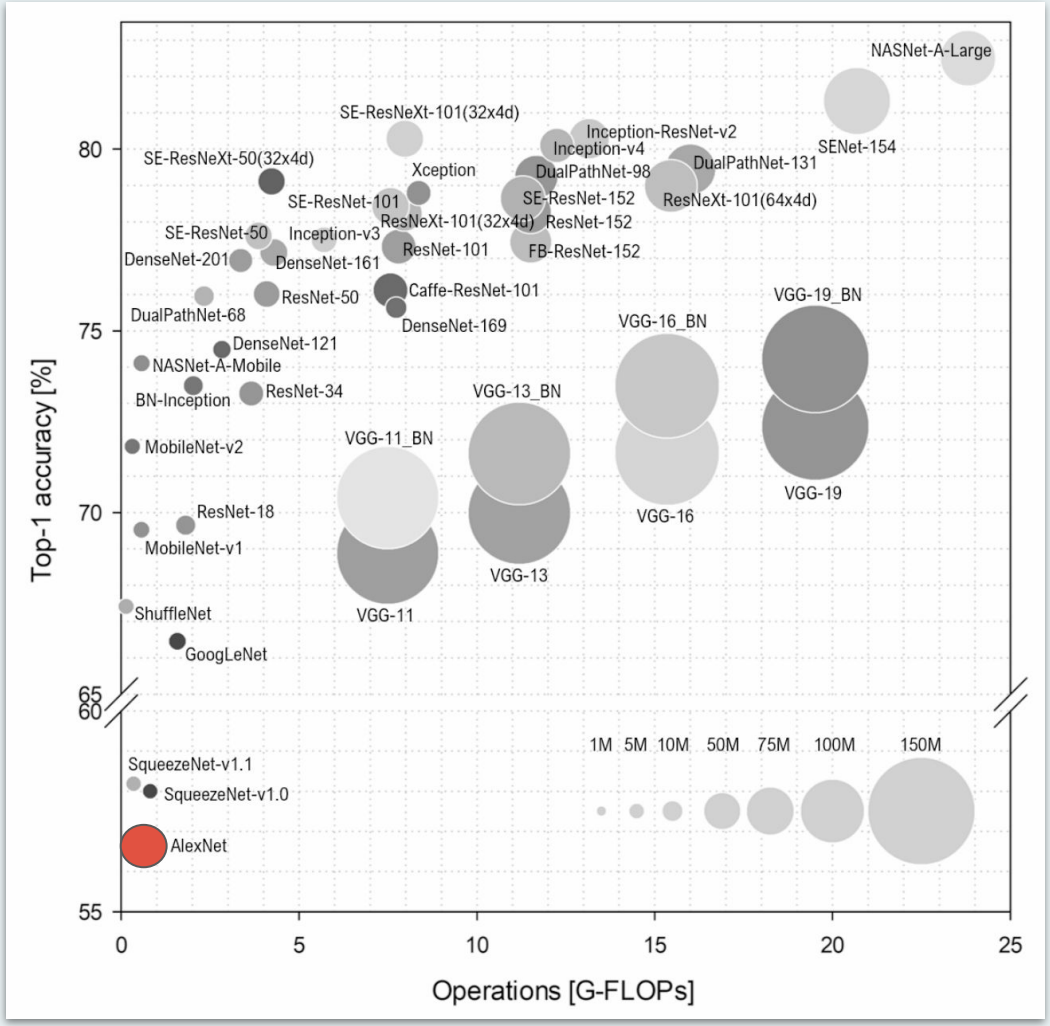
TinyML



ML Model Evolution



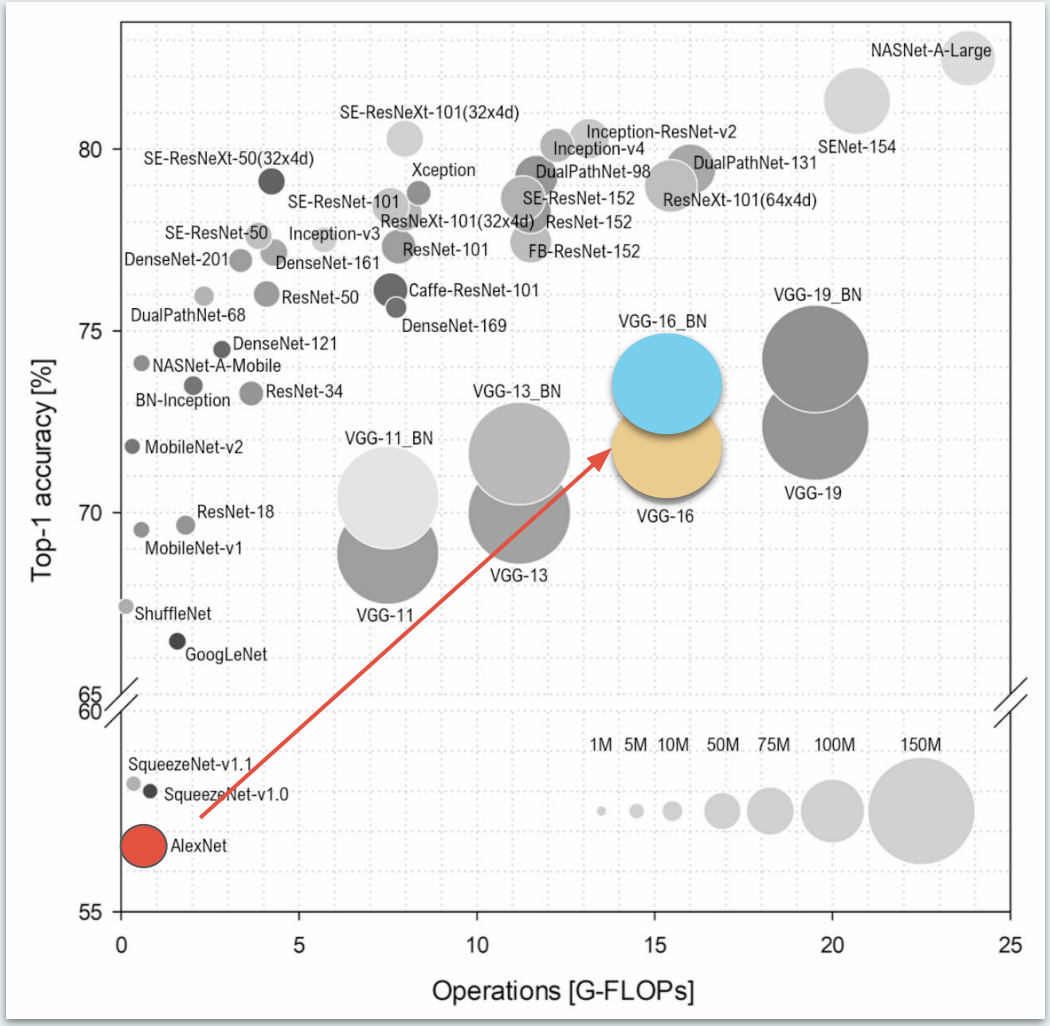
Source: S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures". IEEE Access, vol. 6, 2018.



ML Model Evolution

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

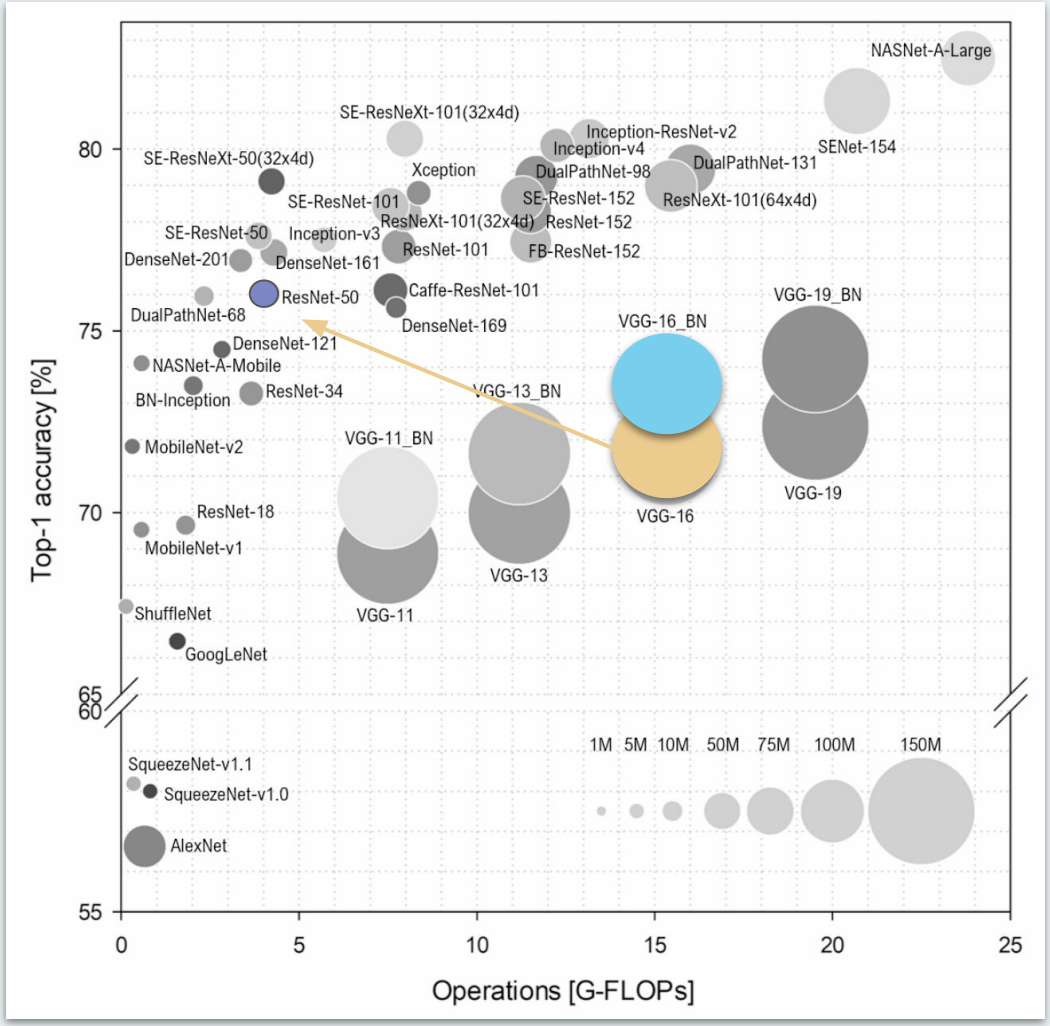
Source: S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures". IEEE Access, vol. 6, 2018.



ML Model Evolution

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

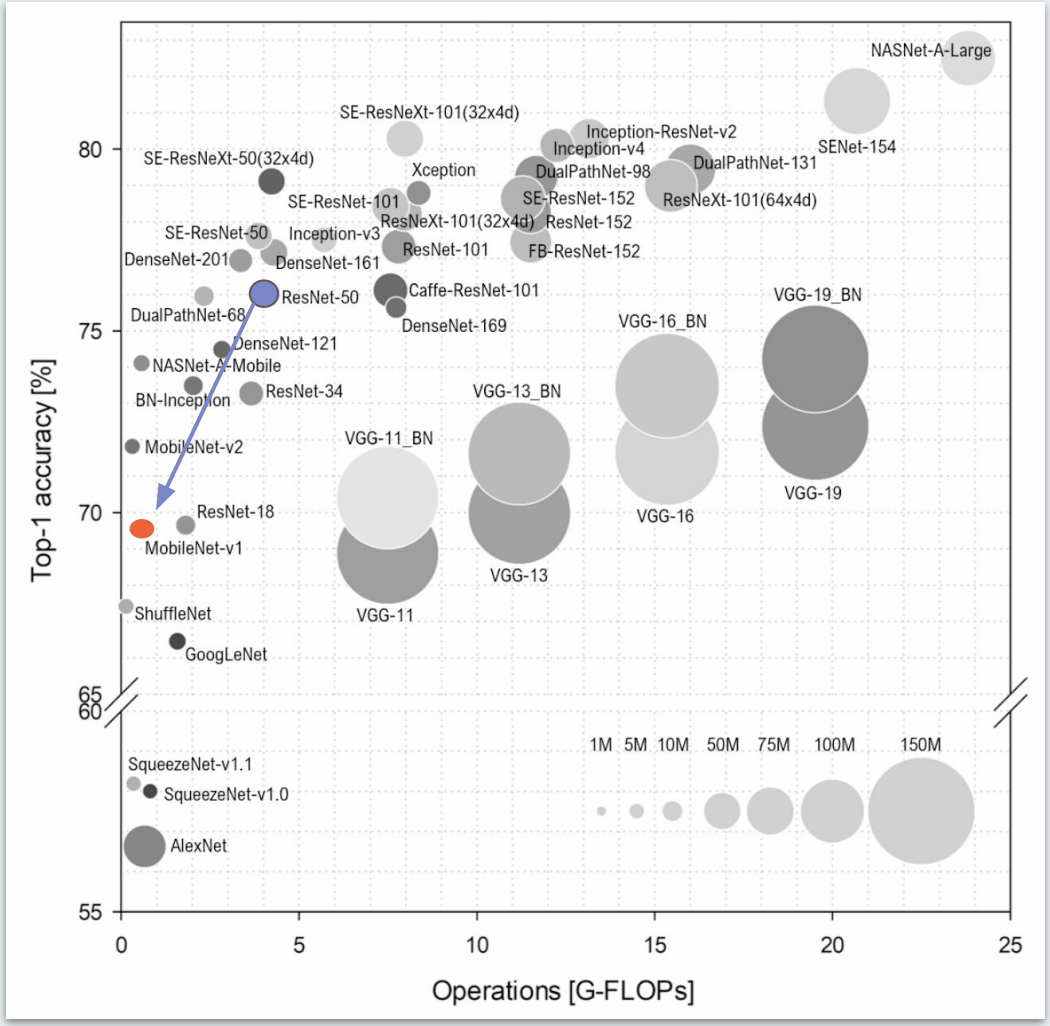
Source: S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures". IEEE Access, vol. 6, 2018.



ML Model Evolution

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

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



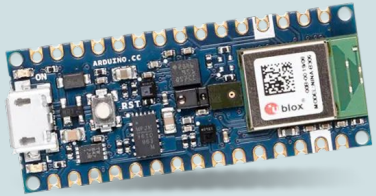
ML Model Evolution

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

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

Problem:

Our board only has 256 KB of RAM (memory) yet MobileNetv1 needs 16.9MB!!!



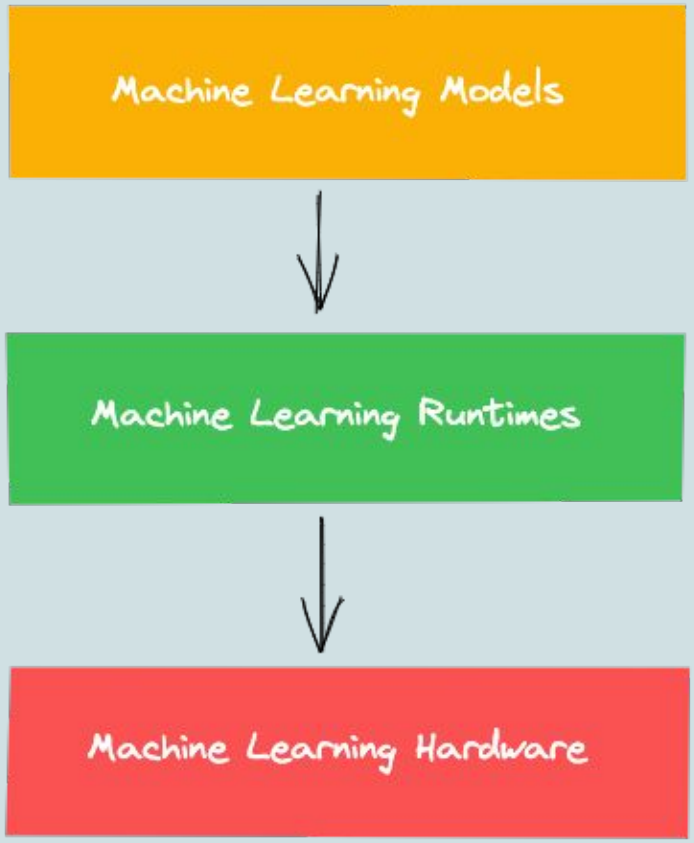
ML Model Evolution

MobileNet (2015)

- 70.6% accuracy
- 16.9 MB in size

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

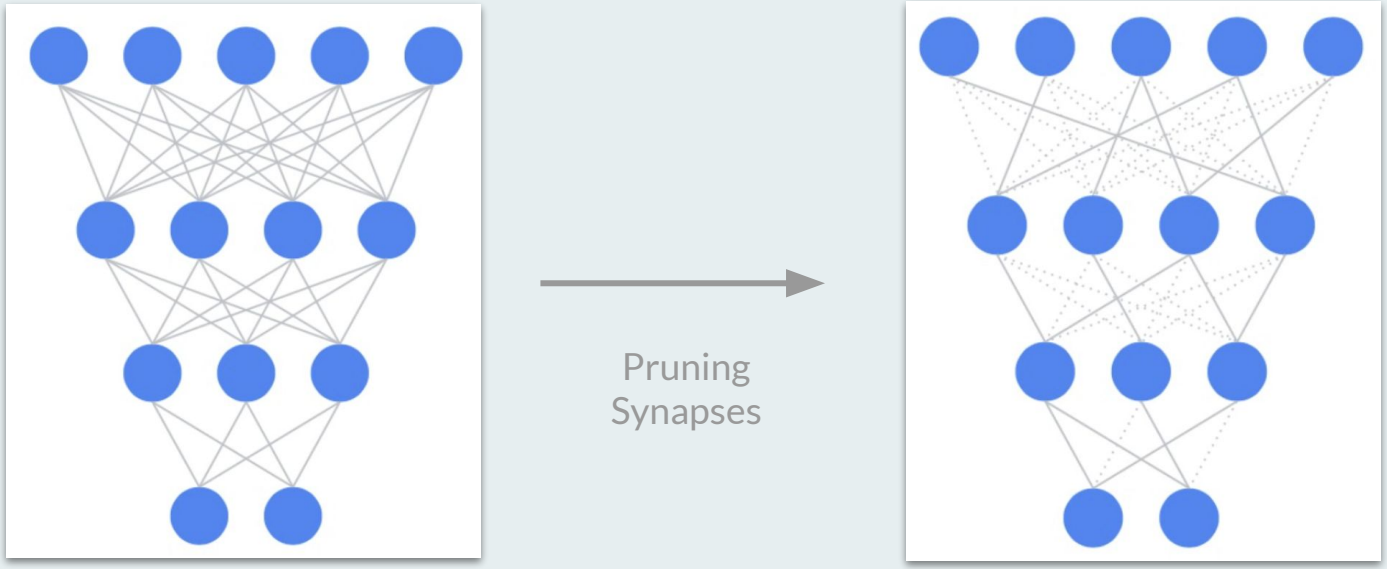
What are the Challenges for TinyML?



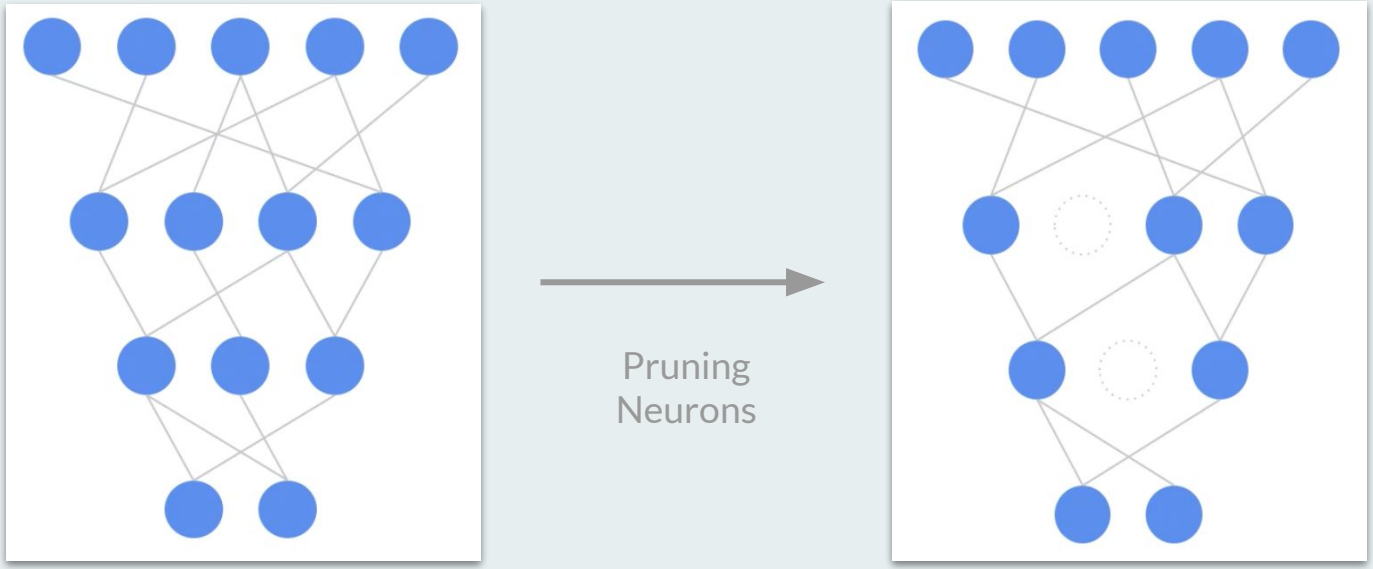
Model Compression Techniques

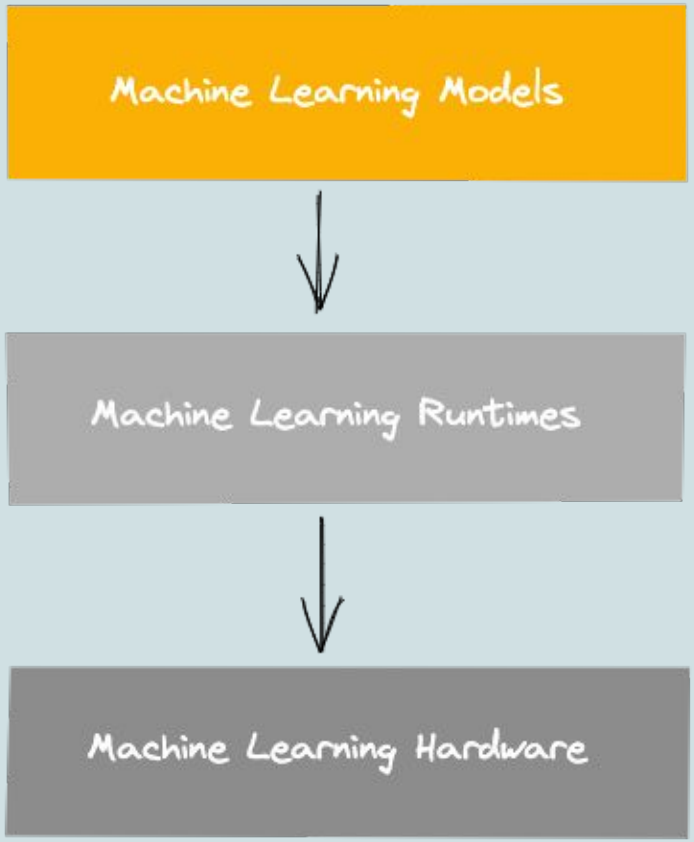
Pruning
Quantization
Knowledge Distillation
...

Pruning



Pruning

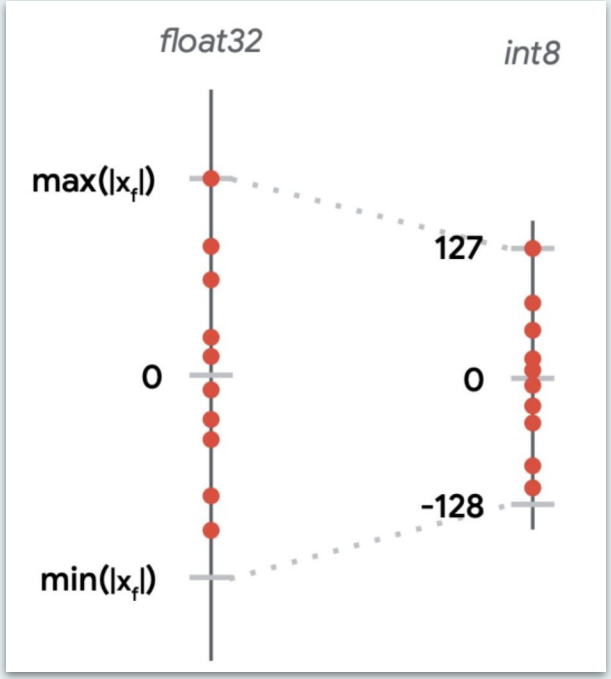


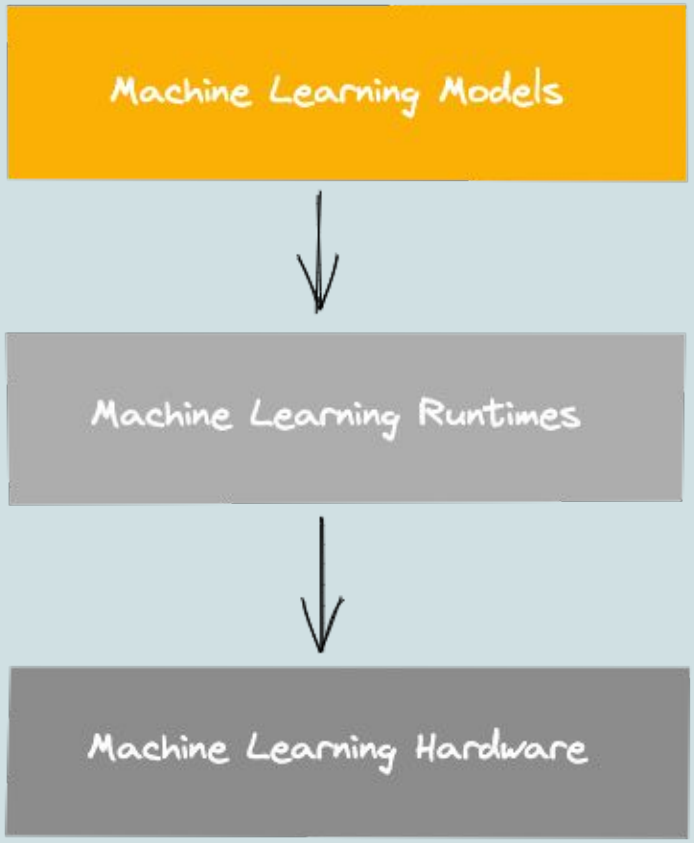


Model Compression Techniques

Pruning
Quantization
Knowledge Distillation
...

Quantization

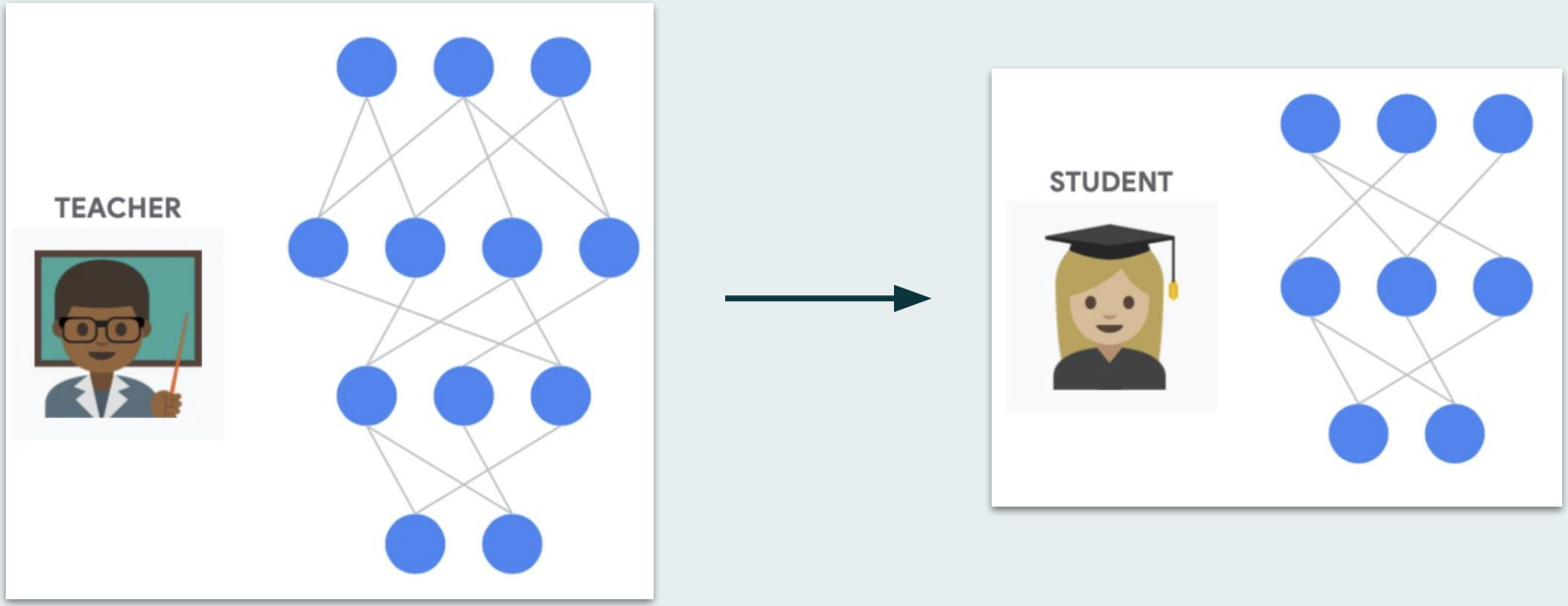


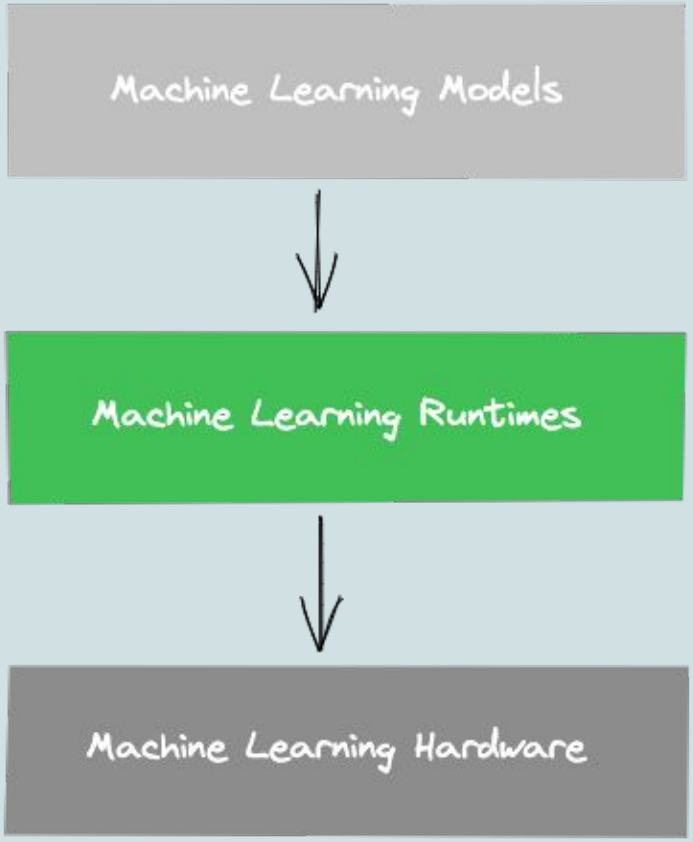


Model Compression Techniques

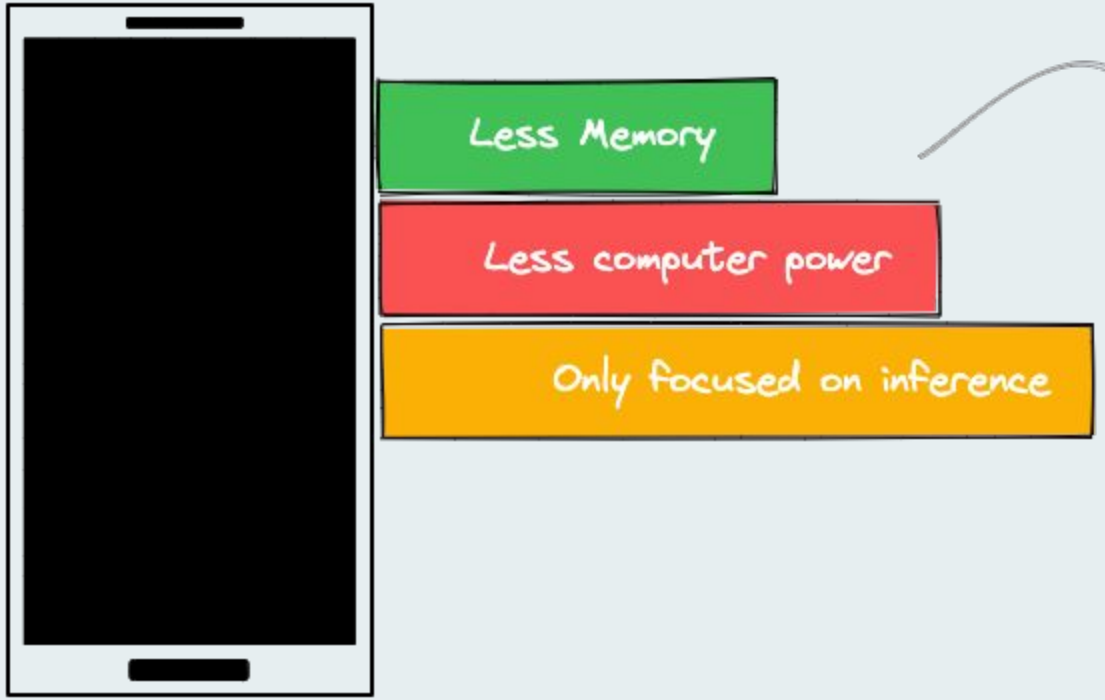
Pruning
Quantization
Knowledge Distillation
...

Knowledge Distillation




Machine Learning Models
Machine Learning Runtimes
Machine Learning Hardware

TensorFlow

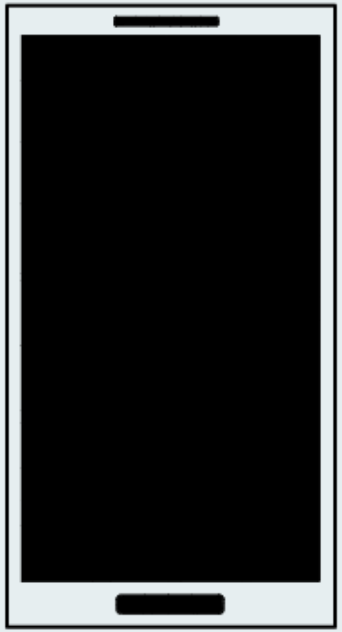


TensorFlow Lite

Key Differences

	TensorFlow	TensorFlow Lite
Topology	Variable	Fixed
Weights	Variable	Fixed
Binary Size	Unimportant	High Priority
Distributed Compute	Needed	Not Needed
Developer Background	ML Researcher	Application Developer

Architecture



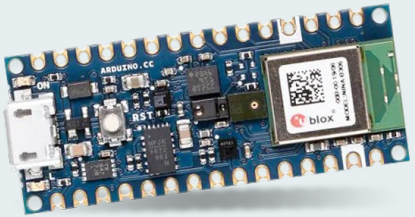
Trained TensorFlow Model

TensorFlow Lite Converter

TensorFlow Lite Model File (.tflite)

iOS

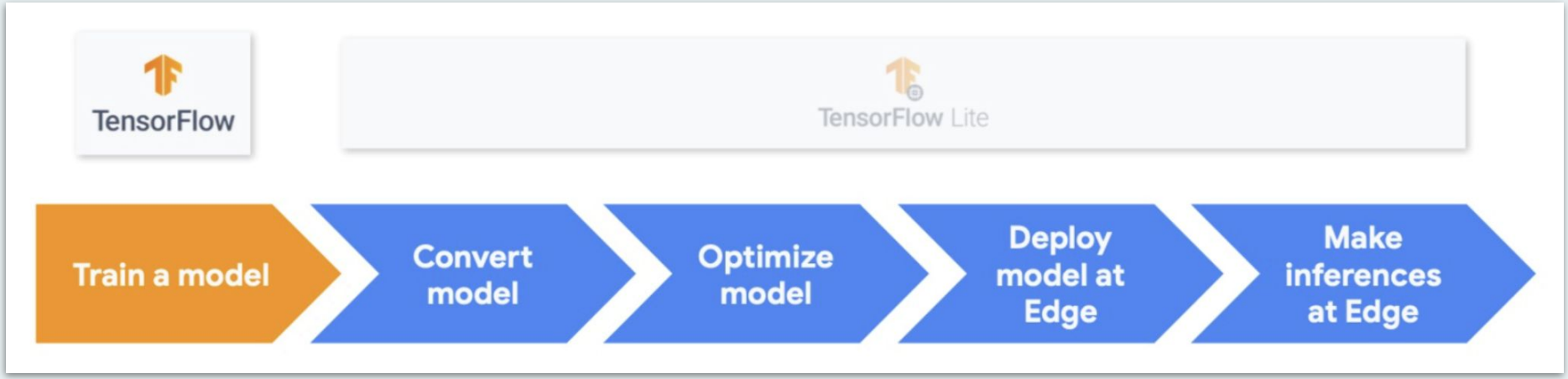
android

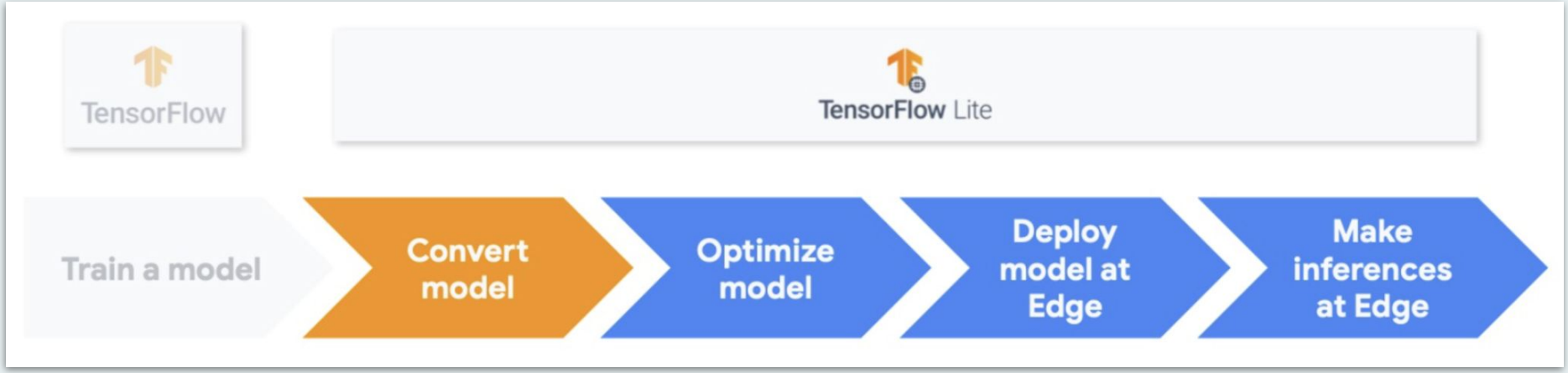


Even less memory

Even less computer power

Also, only focused on inference

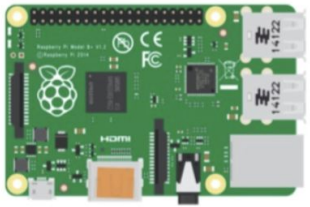

TensorFlow
TensorFlow Lite
Train a model
Convert model
Optimize model
Deploy model at Edge
Make inferences at Edge


TensorFlow
TensorFlow Lite
Train a model
Convert model
Optimize model
Deploy model at Edge
Make inferences at Edge

TensorFlow

TensorFlow Lite

Train a model → Convert model → Optimize model → Deploy model at Edge → Make inferences at Edge



Raspberry Pi

Linux

iOS

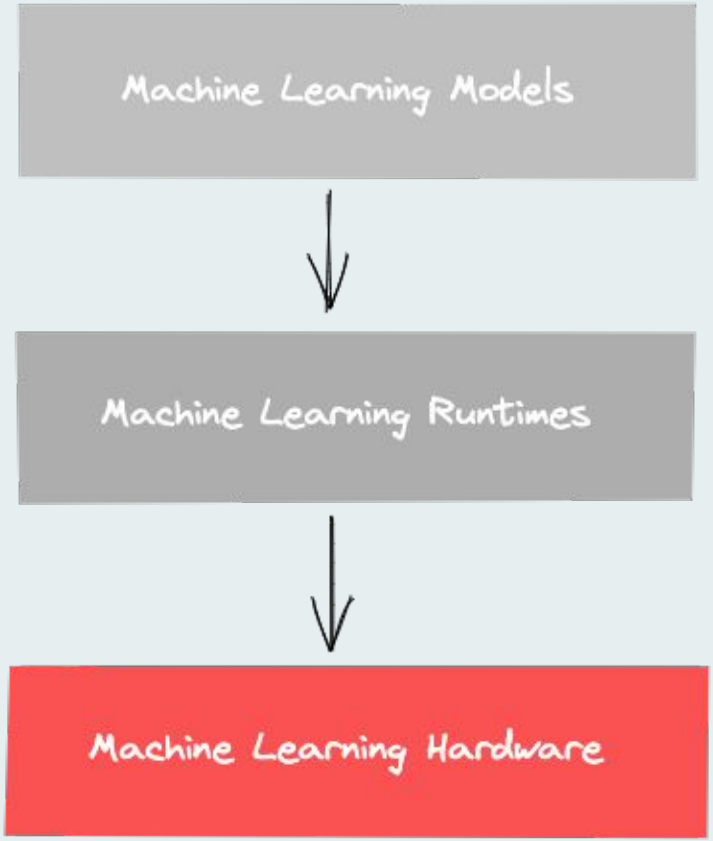
(TF Micro)

Microcontroller

TensorFlow

TensorFlow Lite

Train a model → Convert model → Optimize model → Deploy model at Edge → **Make inferences at Edge**


Machine Learning Models
Machine Learning Runtimes
Machine Learning Hardware