EECE-4710 "IoT and TinyML"

ML Applications, ML Lifecycle, ML Workflow, TensorFlow Lite (TFL) & TFL Micro, TFL Micro HelloWorld Example

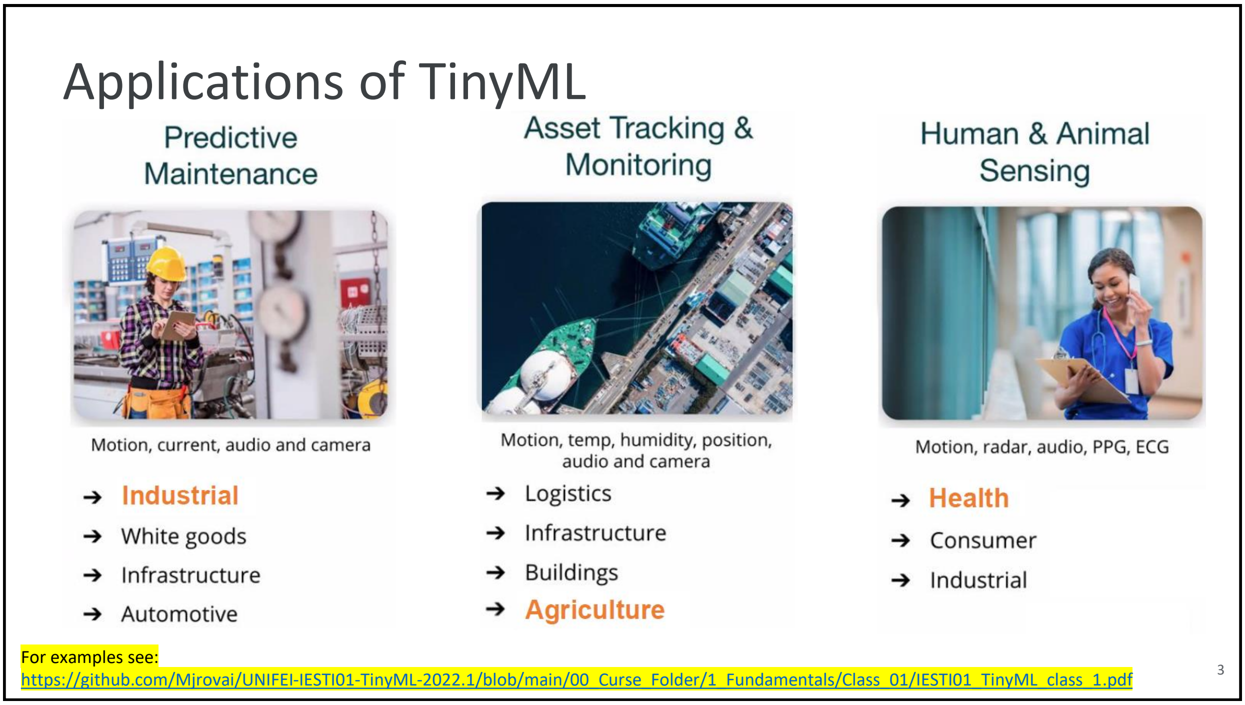
Cris Ababei

MARQUETTE
UNIVERSITY

BE THE DIFFERENCE.

Tiny ML Applications

Examples (Classification, Regression)



Applications of TinyML

Predictive Maintenance

Motion, current, audio and camera

- ➔ **Industrial**
- ➔ White goods
- ➔ Infrastructure
- ➔ Automotive

Asset Tracking & Monitoring

Motion, temp, humidity, position, audio and camera

- ➔ Logistics
- ➔ Infrastructure
- ➔ Buildings
- ➔ **Agriculture**

Human & Animal Sensing

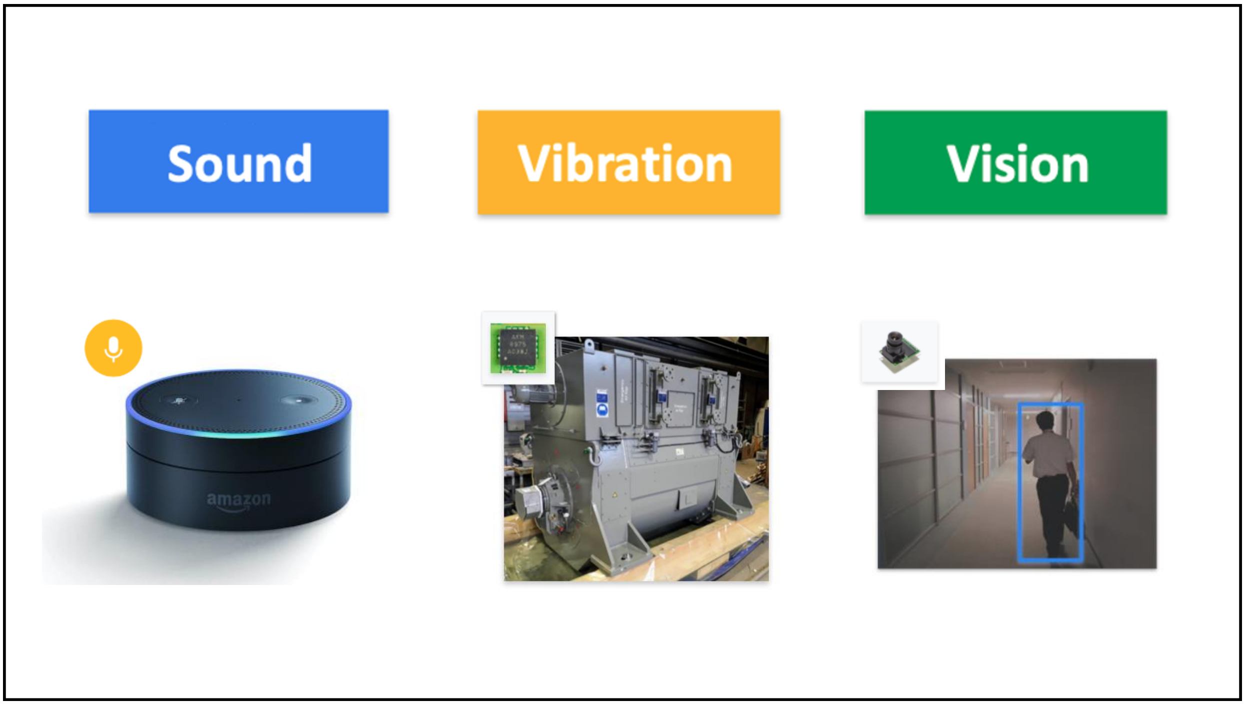
Motion, radar, audio, PPG, ECG

- ➔ **Health**
- ➔ Consumer
- ➔ Industrial

For examples see:
https://github.com/Mjrovai/UNIFEI-IESTI01-TinyML-2022.1/blob/main/00_Curse_Folder/1_Fundamentals/Class_01/IESTI01_TinyML_class_1.pdf

3

3

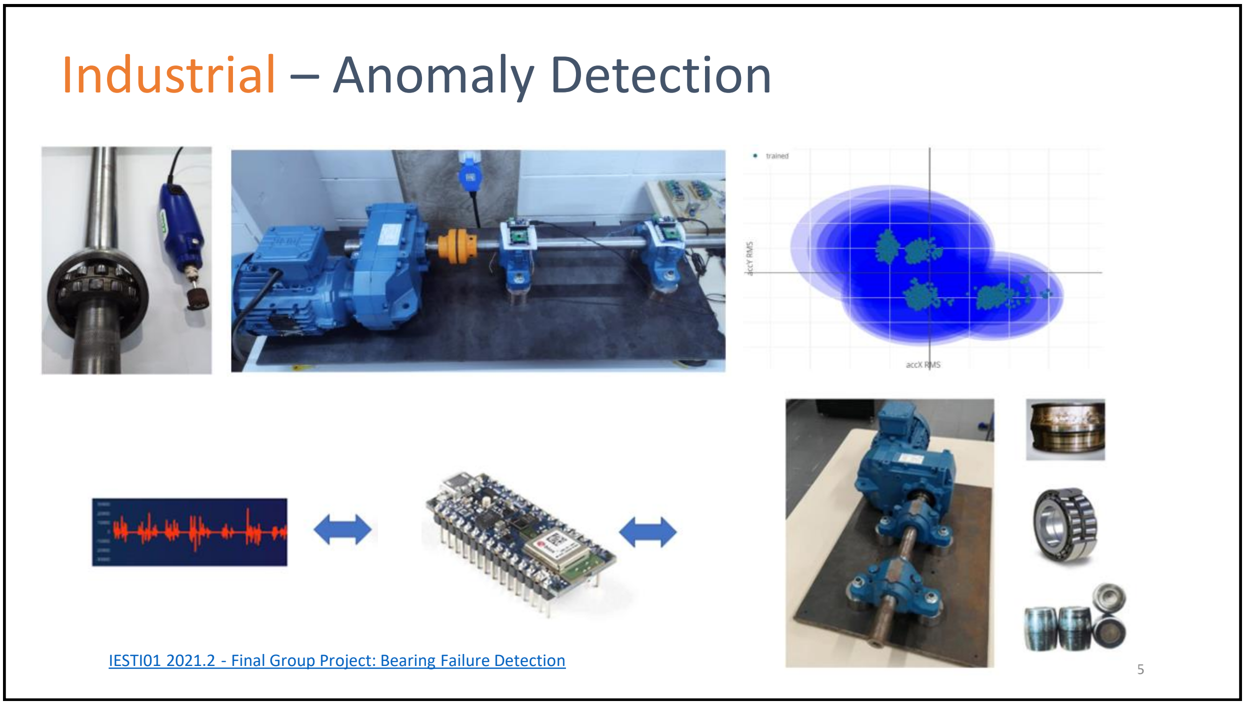


Sound

Vibration

Vision

4

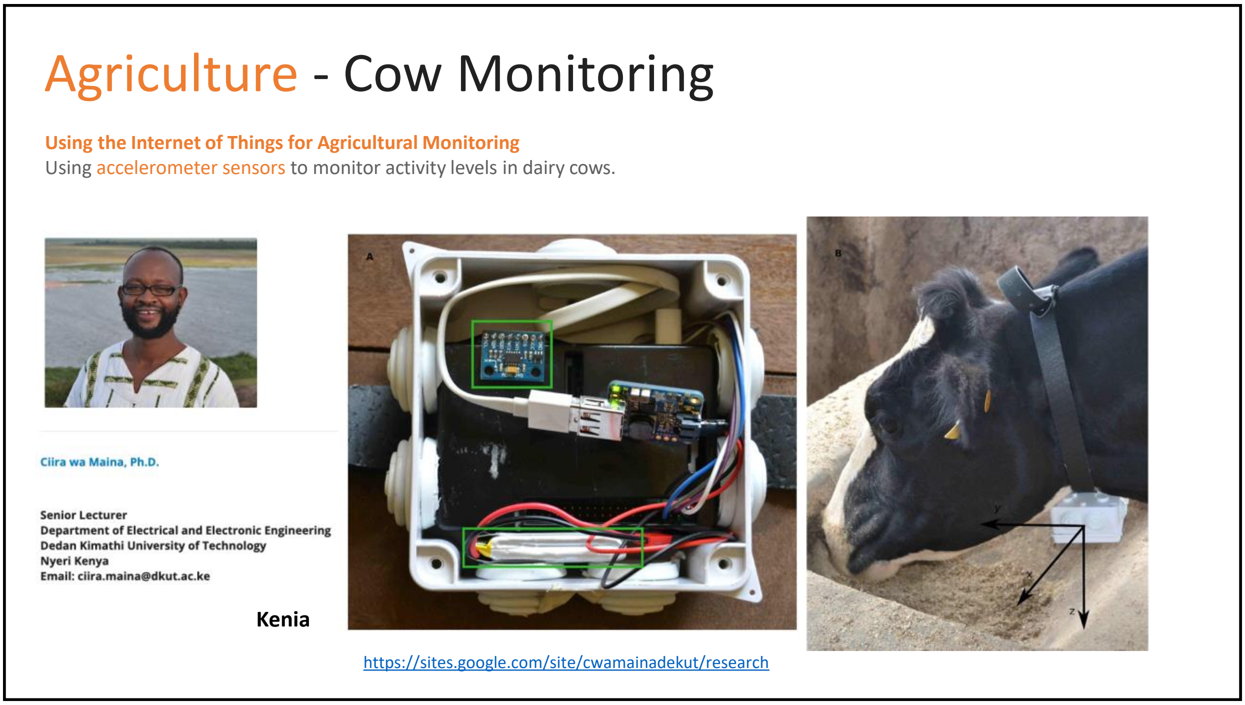


Industrial – Anomaly Detection

IESTI01 2021.2 - Final Group Project: Bearing Failure Detection

5

5



Agriculture - Cow Monitoring

Using the Internet of Things for Agricultural Monitoring

Using accelerometer sensors to monitor activity levels in dairy cows.

Ciira wa Maina, Ph.D.

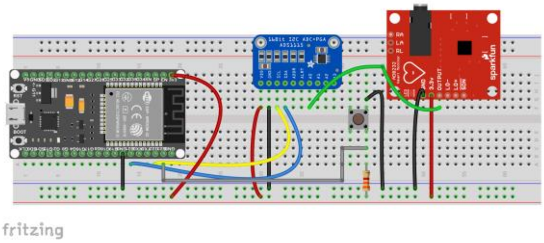
Senior Lecturer
Department of Electrical and Electronic Engineering
Dedan Kimathi University of Technology
Nyeri Kenya
Email: ciira.maina@dkut.ac.ke

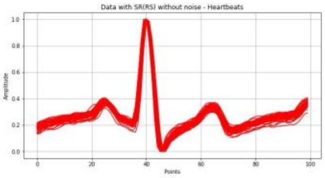
Kenia

https://sites.google.com/site/cwamainadekut/research

6

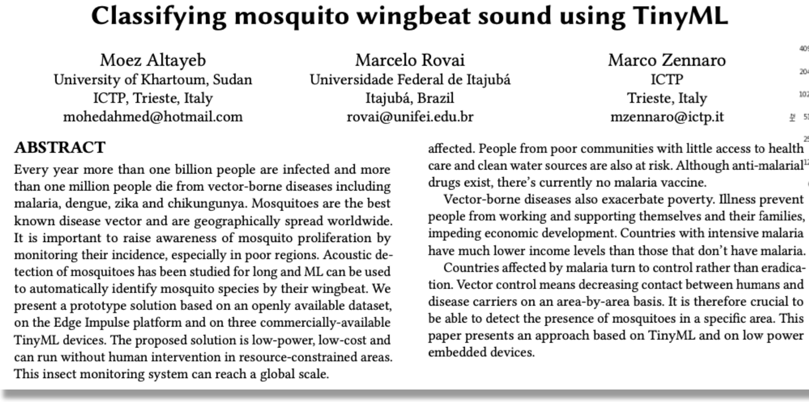
Health - Human Sensing





Guilherme Silva
Engenheiro - UNIFEI

[Atrial Fibrillation Detection on ECG using TinyML](#)
[Silva et al. UNIFEI 2021](#)



Classifying mosquito wingbeat sound using TinyML

Moez Altayeb
University of Khartoum, Sudan
ICTP, Trieste, Italy
mohedahmed@hotmail.com

Marcelo Rovai
Universidade Federal de Itajubá
Itajubá, Brazil
rovai@unifei.edu.br

Marco Zennaro
ICTP
Trieste, Italy
mzennaro@ictp.it

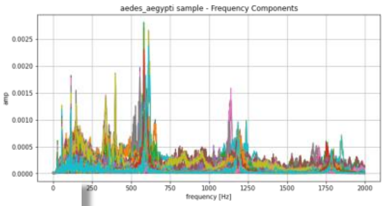
ABSTRACT

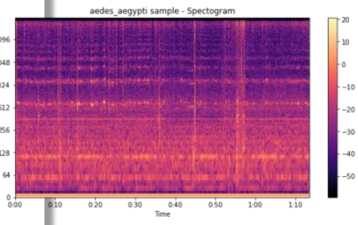
Every year more than one billion people are infected and more than one million people die from vector-borne diseases including malaria, dengue, zika and chikungunya. Mosquitoes are the best known disease vector and are geographically spread worldwide. It is important to raise awareness of mosquito proliferation by monitoring their incidence, especially in poor regions. Acoustic detection of mosquitoes has been studied for long and ML can be used to automatically identify mosquito species by their wingbeat. We present a prototype solution based on an openly available dataset, on the Edge Impulse platform and on three commercially-available TinyML devices. The proposed solution is low-power, low-cost and can run without human intervention in resource-constrained areas. This insect monitoring system can reach a global scale.

affected. People from poor communities with little access to health care and clean water sources are also at risk. Although anti-malarial drugs exist, there's currently no malaria vaccine.

Vector-borne diseases also exacerbate poverty. Illness prevent people from working and supporting themselves and their families, impeding economic development. Countries with intensive malaria have much lower income levels than those that don't have malaria.

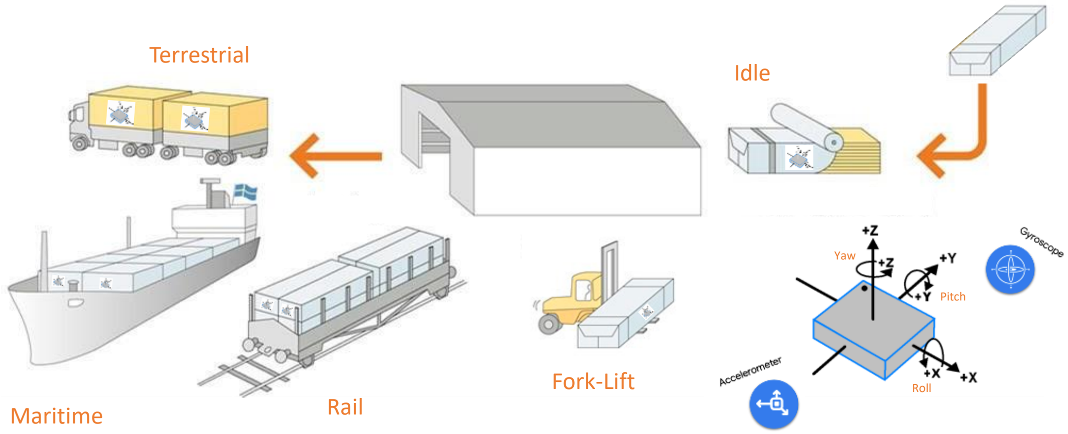
Countries affected by malaria turn to control rather than eradication. Vector control means decreasing contact between humans and disease carriers on an area-by-area basis. It is therefore crucial to be able to detect the presence of mosquitoes in a specific area. This paper presents an approach based on TinyML and on low power embedded devices.





https://github.com/Mjrovai/wingbeat-mosquito-tinyml

Mechanical Stresses in Transport



ICTP SciTyniML 21 - Hands on Embedded ML - Motion/Anomaly Detection and Scientific Applications

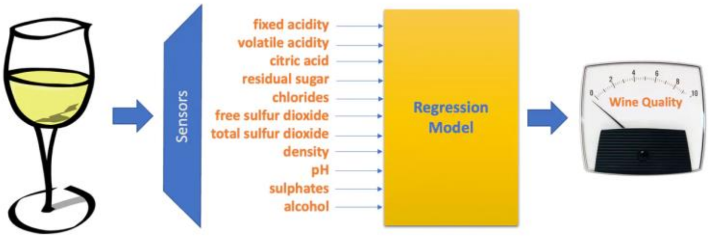
Coffee Disease Classification

João Vitor Yukio Bordin Yamashita
Graduando em Engenharia Eletrônica pela UNIFEI

https://www.hackster.io/Yukio/coffee-disease-classification-with-ml-b0a3fc

[Estimate Weight From a Photo Using Visual Regression in Edge Impulse]

Regression on TinyML

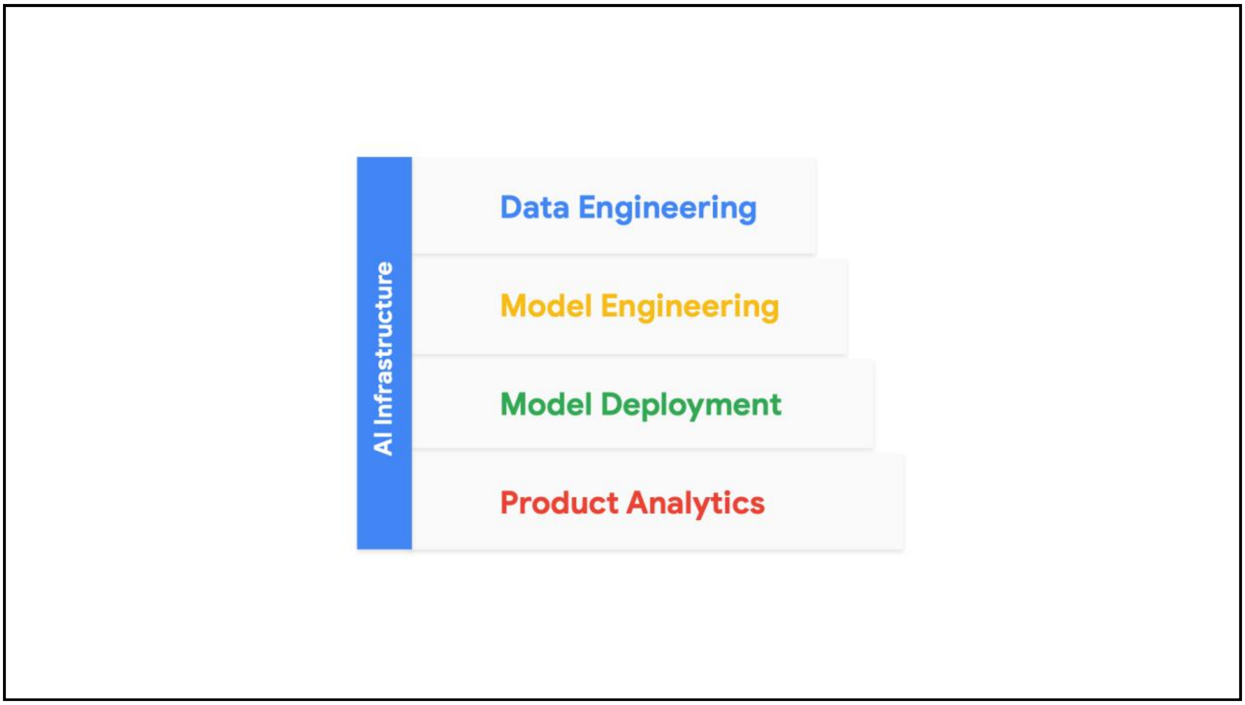


[TinyML Made Easy: Exploring Regression - White Wine Quality]

11

ML Lifecycle

12



AI Infrastructure

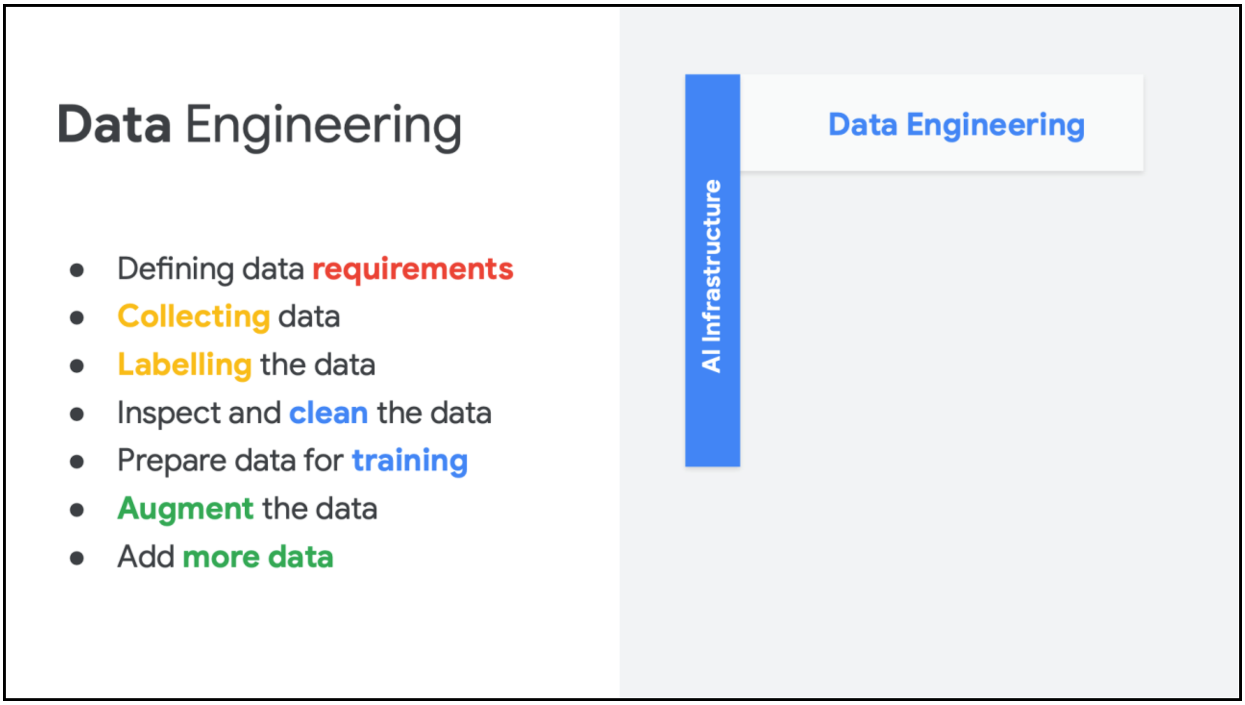
Data Engineering

Model Engineering

Model Deployment

Product Analytics

13



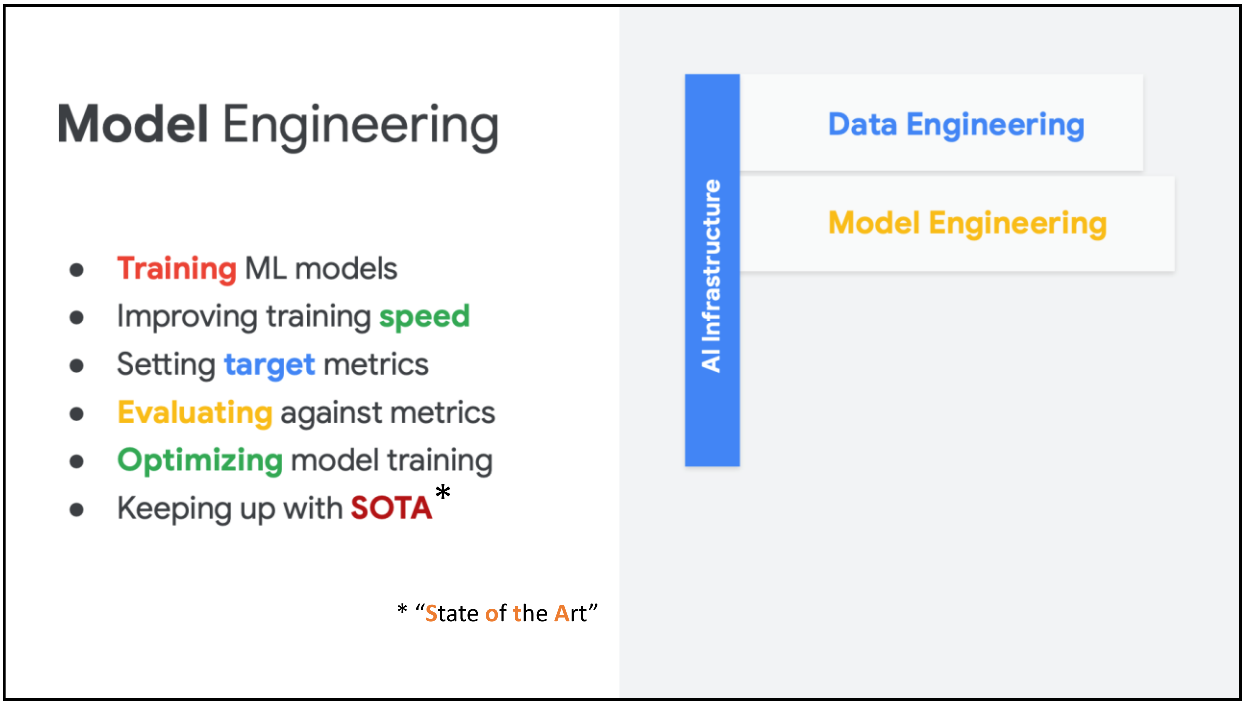
Data Engineering

- Defining data **requirements**
- **Collecting** data
- **Labelling** the data
- Inspect and **clean** the data
- Prepare data for **training**
- **Augment** the data
- Add **more data**

AI Infrastructure

Data Engineering

14



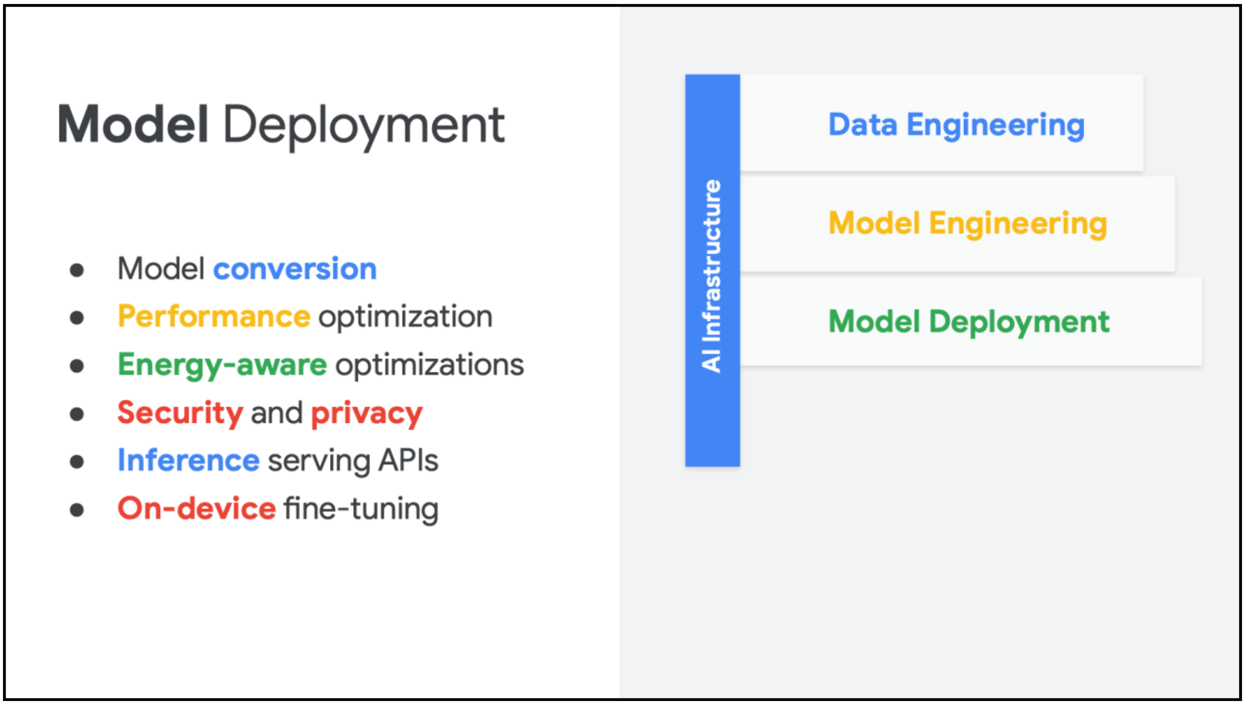
Model Engineering

- **Training** ML models
- Improving training **speed**
- Setting **target** metrics
- **Evaluating** against metrics
- **Optimizing** model training
- Keeping up with **SOTA***

AI Infrastructure
Data Engineering
Model Engineering

* "State of the Art"

15

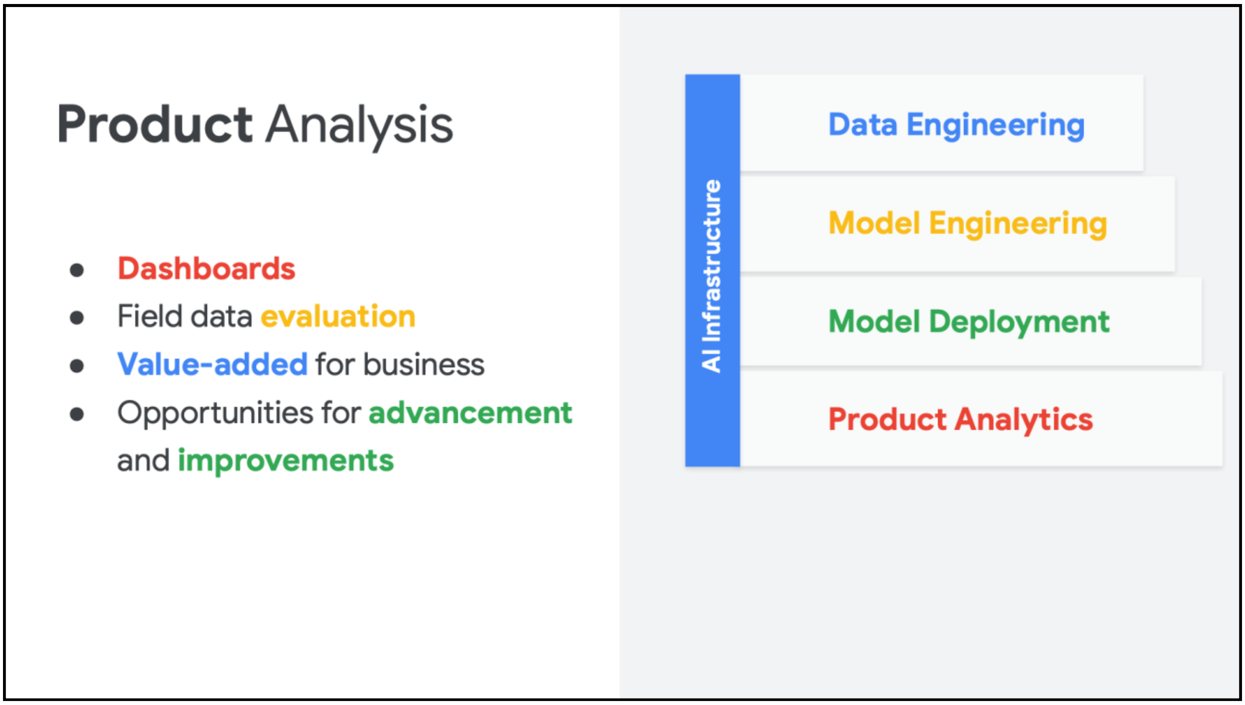


Model Deployment

- Model **conversion**
- **Performance** optimization
- **Energy-aware** optimizations
- **Security** and **privacy**
- **Inference** serving APIs
- **On-device** fine-tuning

AI Infrastructure
Data Engineering
Model Engineering
Model Deployment

16



Product Analysis

- Dashboards
- Field data evaluation
- Value-added for business
- Opportunities for advancement and improvements

AI Infrastructure

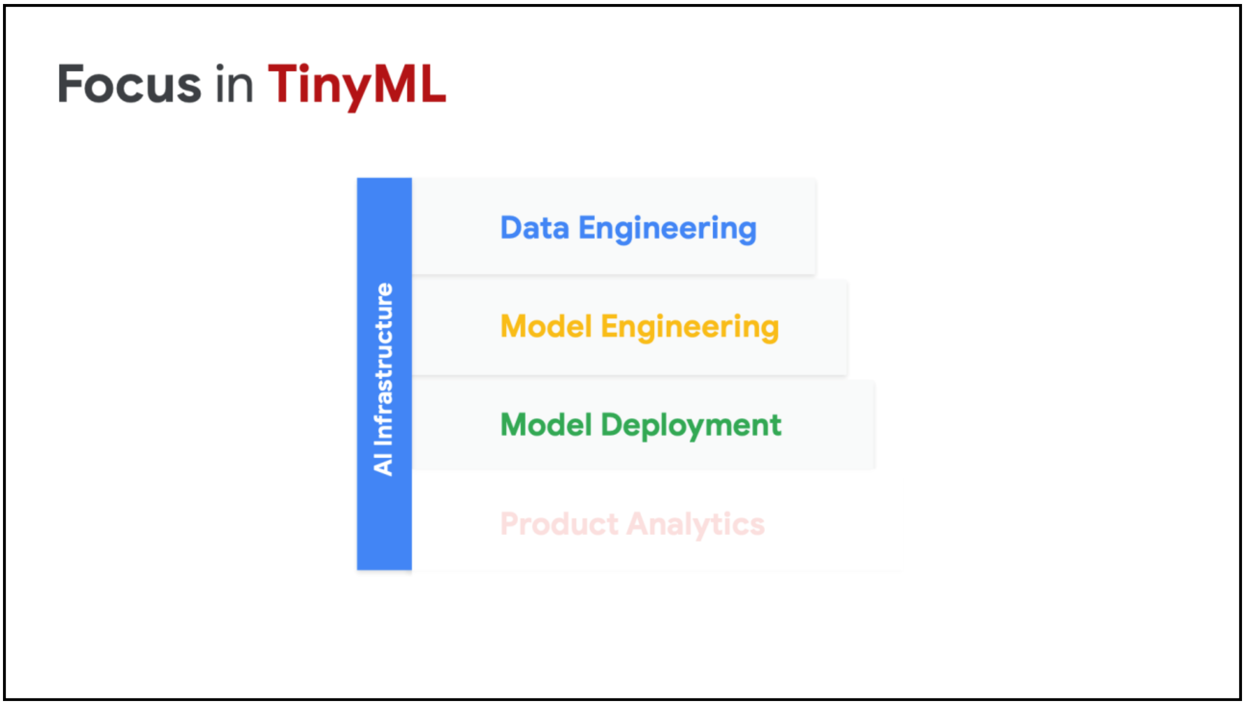
Data Engineering

Model Engineering

Model Deployment

Product Analytics

17



Focus in TinyML

AI Infrastructure

Data Engineering

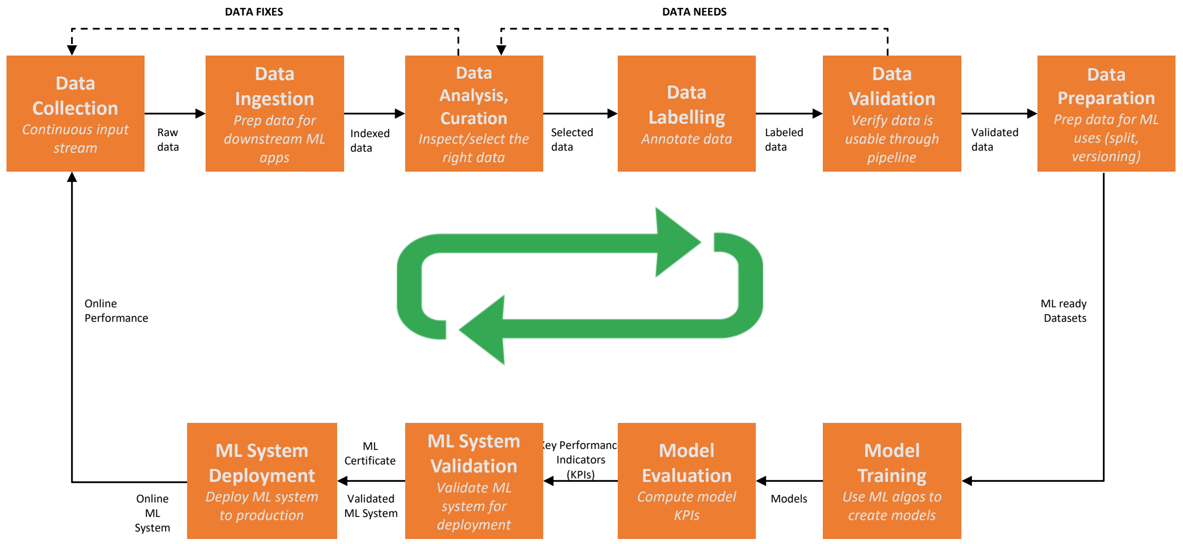
Model Engineering

Model Deployment

Product Analytics

18

Life-cycle of ML



DATA FIXES

DATA NEEDS

Data Collection *Continuous input stream*

Raw data

Data Ingestion *Prep data for downstream ML apps*

Indexed data

Data Analysis, Curation *Inspect/select the right data*

Selected data

Data Labelling *Annotate data*

Labeled data

Data Validation *Verify data is usable through pipeline*

Validated data

Data Preparation *Prep data for ML uses (split, versioning)*

ML ready Datasets

Model Training *Use ML algos to create models*

Models

Model Evaluation *Compute model KPIs*

Key Performanc Indicators (KPIs)

ML System Validation *Validate ML system for deployment*

ML Certificate

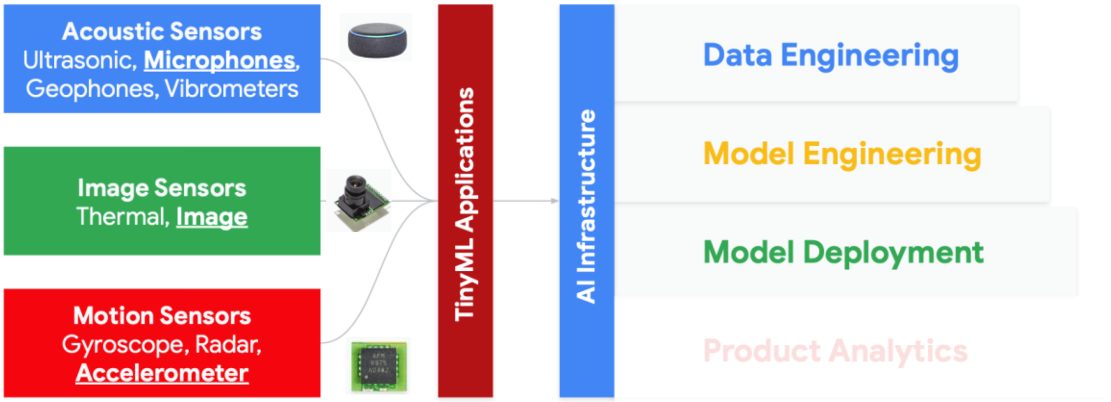
Validated ML System

ML System Deployment *Deploy ML system to production*

Online ML System

Online Performance

ML Workflow



Acoustic Sensors
Ultrasonic, **Microphones**, Geophones, Vibrometers

Image Sensors
Thermal, **Image**

Motion Sensors
Gyroscope, Radar, **Accelerometer**

TinyML Applications

AI Infrastructure

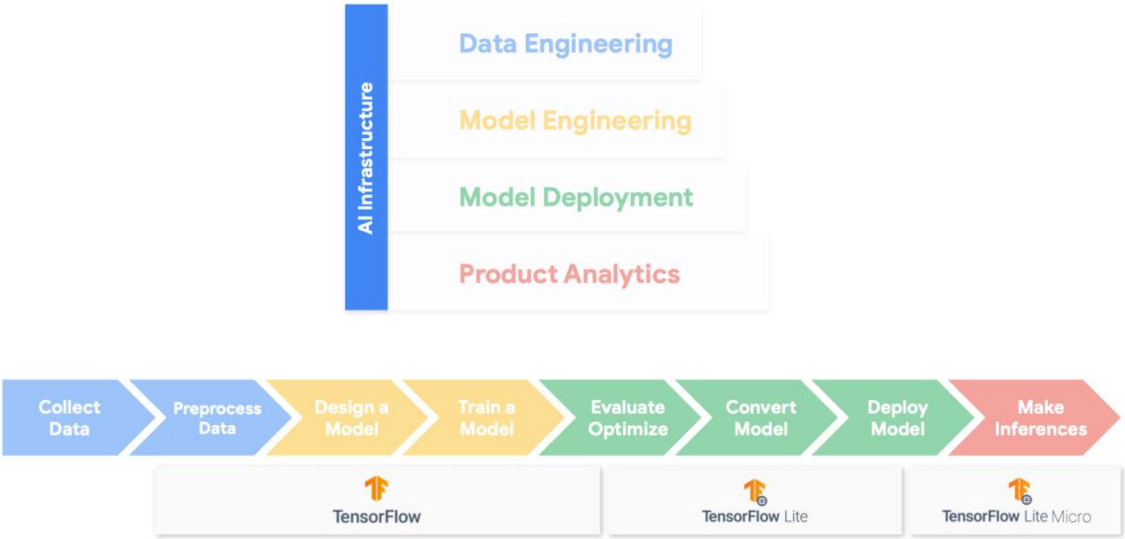
Data Engineering

Model Engineering

Model Deployment

Product Analytics

21



AI Infrastructure

Data Engineering

Model Engineering

Model Deployment

Product Analytics

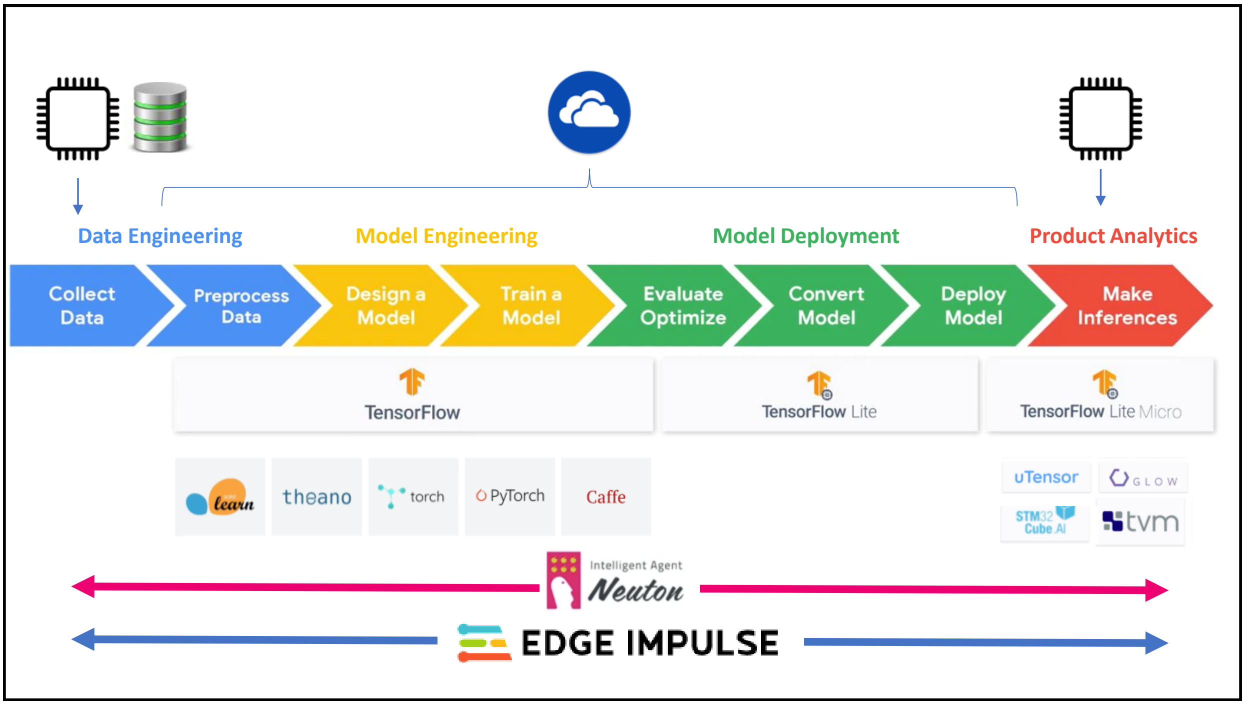
Collect Data → Preprocess Data → Design a Model → Train a Model → Evaluate Optimize → Convert Model → Deploy Model → Make Inferences

TensorFlow

TensorFlow Lite

TensorFlow Lite Micro

22



Data Engineering

Model Engineering

Model Deployment

Product Analytics

Collect Data → Preprocess Data → Design a Model → Train a Model → Evaluate Optimize → Convert Model → Deploy Model → Make Inferences

TensorFlow

TensorFlow Lite

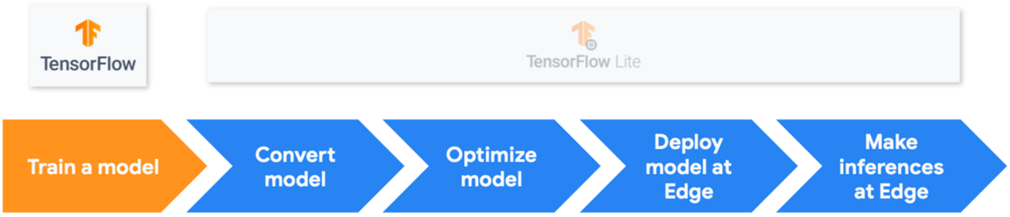
TensorFlow Lite Micro

23

TensorFlow Lite (TFL)

Inference on the Edge

24



TensorFlow

TensorFlow Lite

Train a model

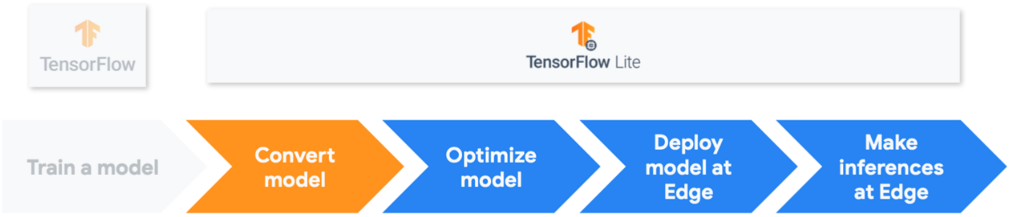
Convert model

Optimize model

Deploy model at Edge

Make inferences at Edge

25



TensorFlow

TensorFlow Lite

Train a model

Convert model

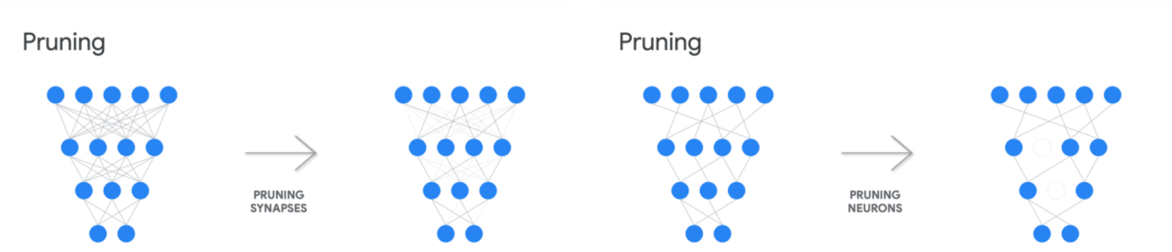
Optimize model

Deploy model at Edge

Make inferences at Edge

26

Pruning



More info: An introduction to weight pruning by Tivadar Danka

27

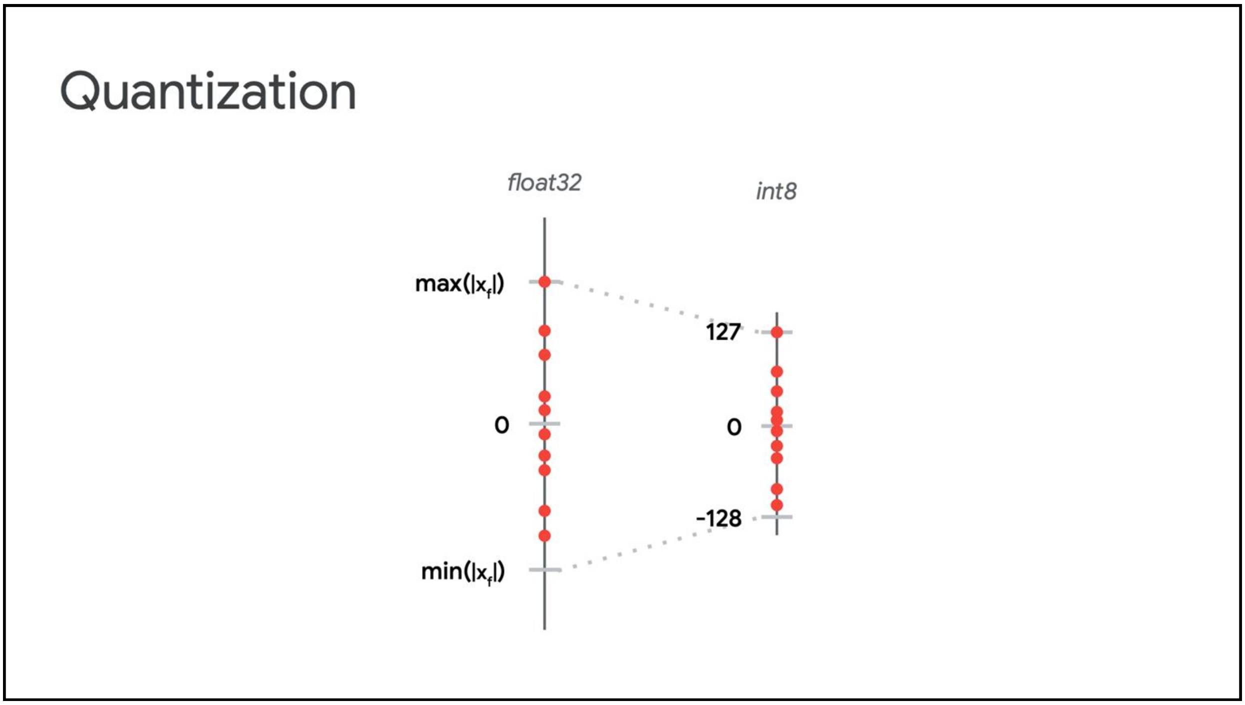
Quantization

Quantization is an optimization that works by **reducing the precision** of the numbers used to represent a model's parameters, which by default are 32-bit floating point numbers. This results in a:

✓ **smaller model size**,
✓ **better portability (*)** and
✓ **faster computation**

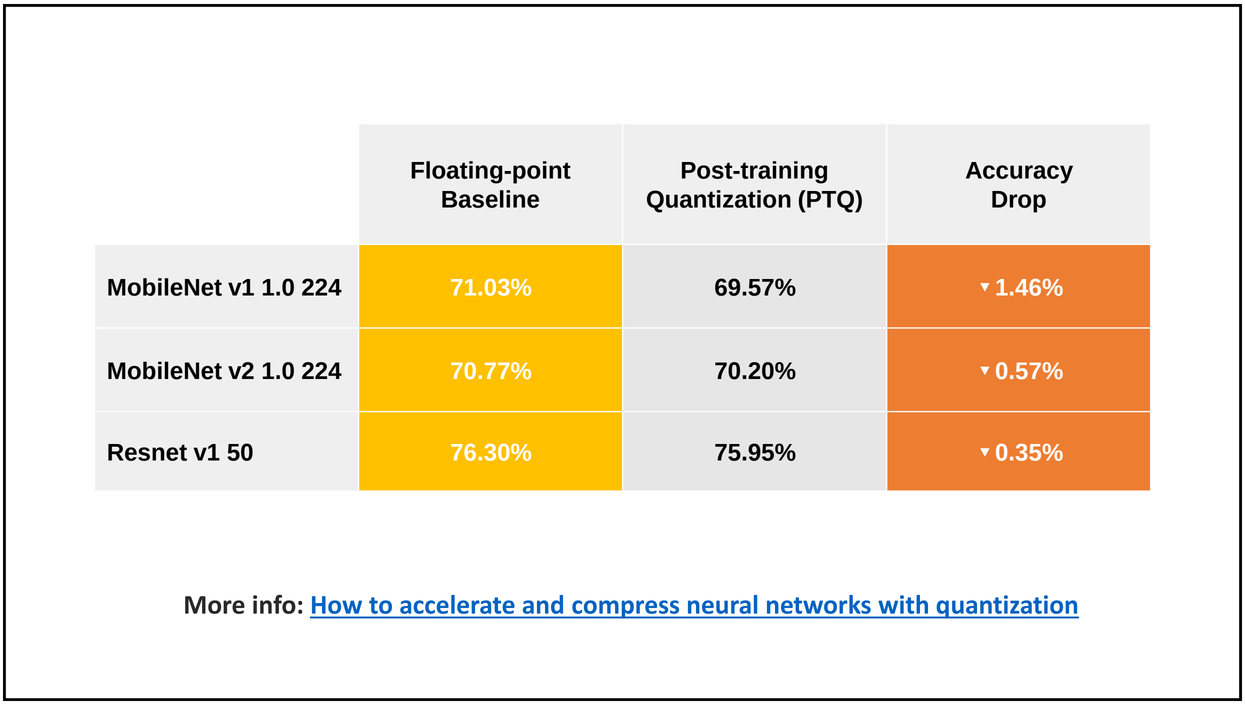
(*) A lot of MCUs do not handle Float-Point operations

28



Quantization

29



	Floating-point Baseline	Post-training Quantization (PTQ)	Accuracy Drop
MobileNet v1 1.0 224	71.03%	69.57%	▼1.46%
MobileNet v2 1.0 224	70.77%	70.20%	▼0.57%
Resnet v1 50	76.30%	75.95%	▼0.35%

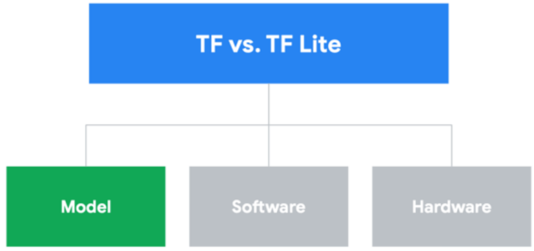
More info: How to accelerate and compress neural networks with quantization

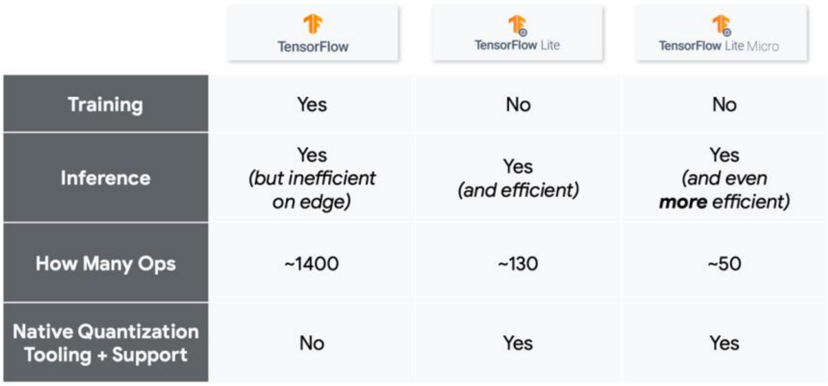
30

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

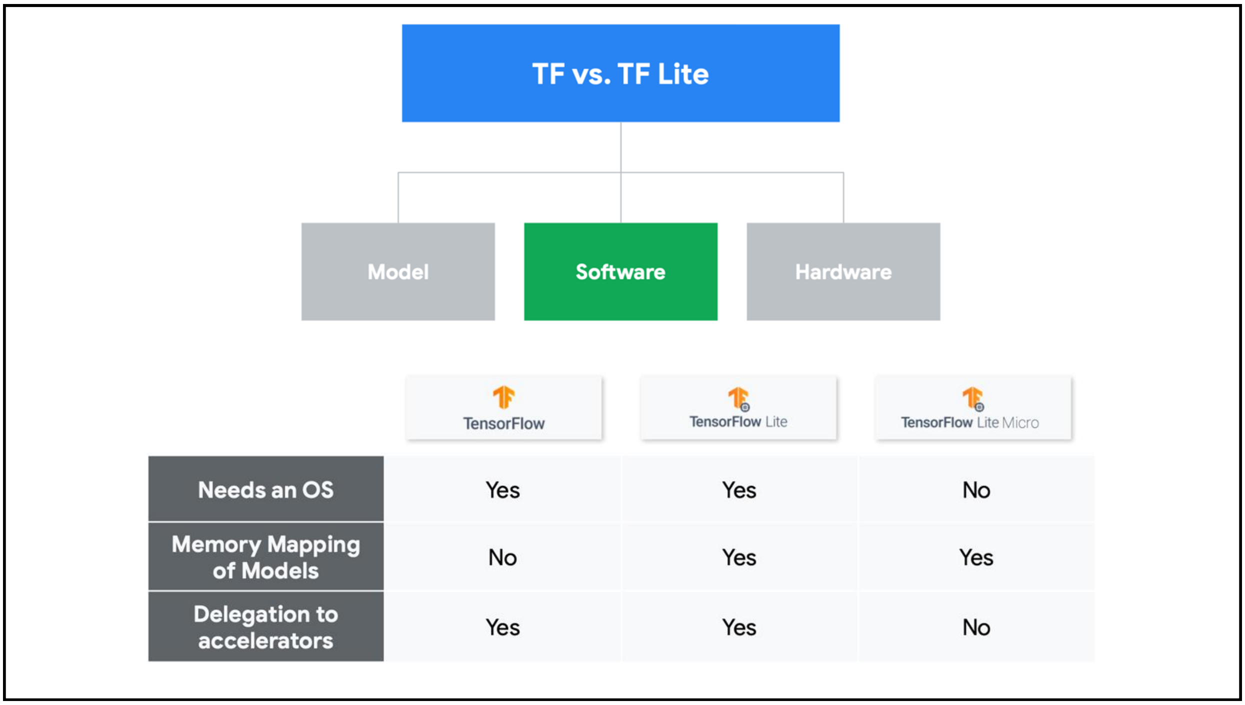
31





	TensorFlow	TensorFlow Lite	TensorFlow Lite Micro
Training	Yes	No	No
Inference	Yes *(but inefficient on edge)*	Yes *(and efficient)*	Yes *(and even **more** efficient)*
How Many Ops	~1400	~130	~50
Native Quantization Tooling + Support	No	Yes	Yes

32

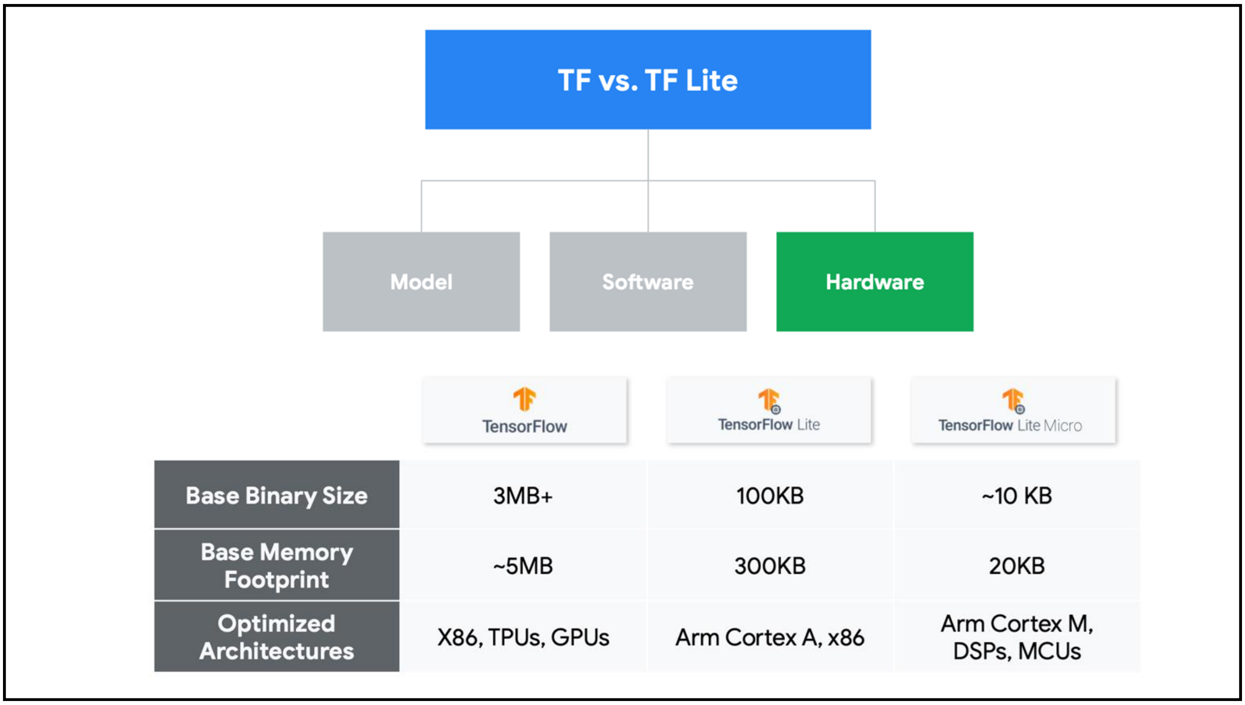


TF vs. TF Lite

Model | Software | Hardware

	TensorFlow	TensorFlow Lite	TensorFlow Lite Micro
Needs an OS	Yes	Yes	No
Memory Mapping of Models	No	Yes	Yes
Delegation to accelerators	Yes	Yes	No

33



TF vs. TF Lite

Model | Software | Hardware

	TensorFlow	TensorFlow Lite	TensorFlow Lite Micro
Base Binary Size	3MB+	100KB	~10 KB
Base Memory Footprint	~5MB	300KB	20KB
Optimized Architectures	X86, TPUs, GPUs	Arm Cortex A, x86	Arm Cortex M, DSPs, MCUs

34

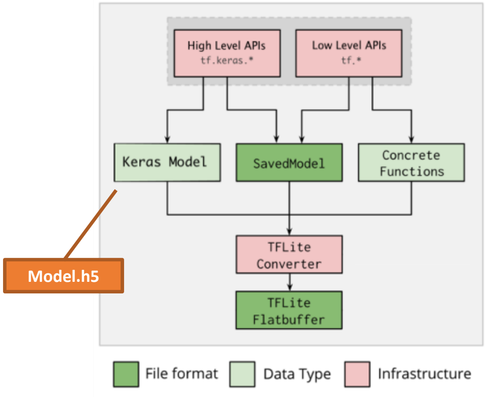
Optimization and Quantization

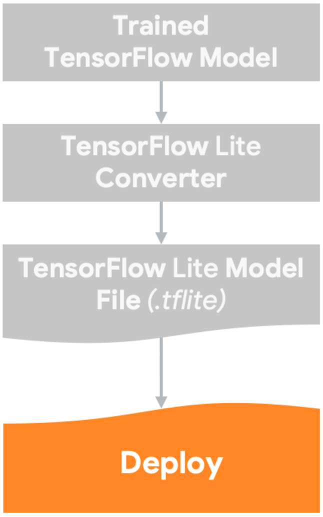
Minimizing compression loss

35

35

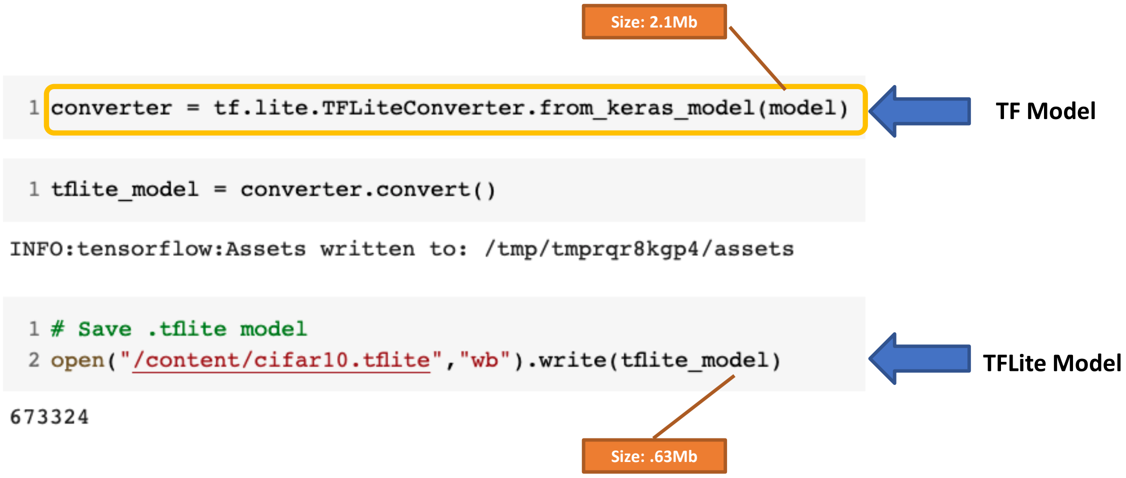
TensorFlow Workflow




Trained TensorFlow Model
TensorFlow Lite Converter
TensorFlow Lite Model File (.tflite)
Deploy

36

Converting TF Model to TFLite Model



```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
```

```
tflite_model = converter.convert()
```

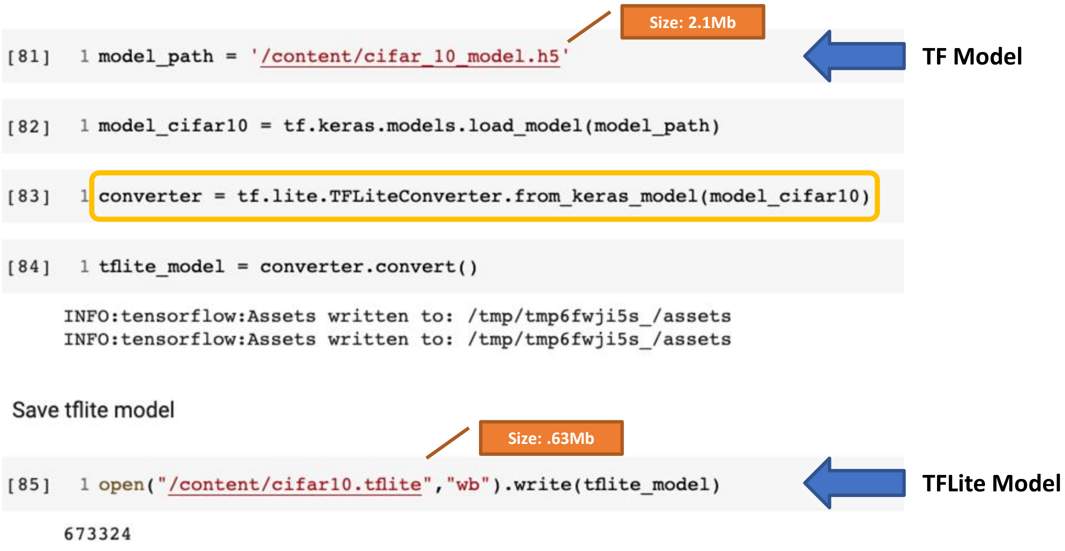
```
INFO:tensorflow:Assets written to: /tmp/tmprqr8kgp4/assets
```

```
# Save .tflite model
open("/content/cifar10.tflite","wb").write(tflite_model)
```

```
673324
```

37

Converting From a Saved Model



```
[81] model_path = '/content/cifar_10_model.h5'
```

```
[82] model_cifar10 = tf.keras.models.load_model(model_path)
```

```
[83] converter = tf.lite.TFLiteConverter.from_keras_model(model_cifar10)
```

```
[84] tflite_model = converter.convert()

     INFO:tensorflow:Assets written to: /tmp/tmp6fwji5s_/assets
     INFO:tensorflow:Assets written to: /tmp/tmp6fwji5s_/assets
```

Save tflite model

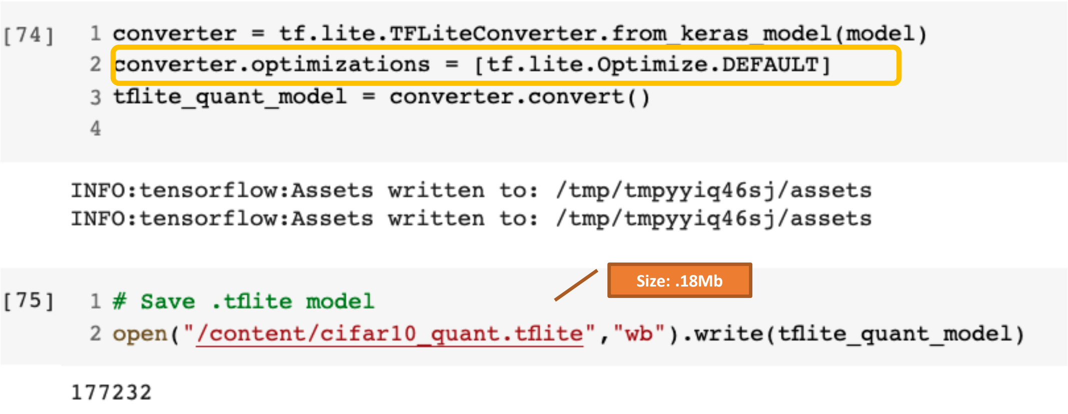
```
[85] open("/content/cifar10.tflite","wb").write(tflite_model)

     673324
```

38

Dynamic range quantization

The simplest form of post-training quantization statically quantizes only the weights from floating point to integer, which has 8-bits of precision:



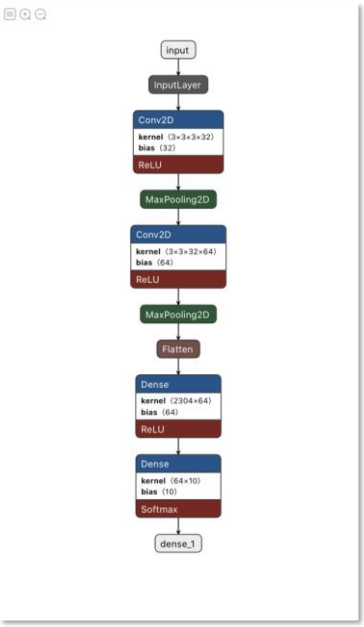
```
[74]
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_quant_model = converter.convert()

INFO:tensorflow:Assets written to: /tmp/tmpyyiq46sj/assets
INFO:tensorflow:Assets written to: /tmp/tmpyyiq46sj/assets
```

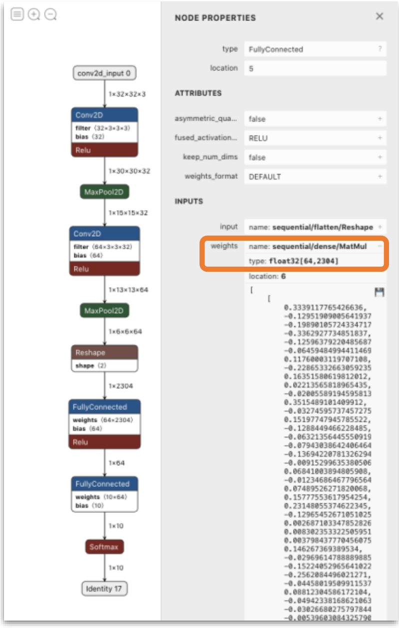
```
[75]
# Save .tflite model
open("/content/cifar10_quant.tflite","wb").write(tflite_quant_model)

177232
```

Size: .18Mb



Cifar_10.h5



NODE PROPERTIES

type: FullyConnected

location: 5

ATTRIBUTES

asymmetric_qua...: false

fused_activation...: RELU

keep_num_dims: false

weights_format: DEFAULT

INPUTS

input: name: **sequential/flatten/Reshape**

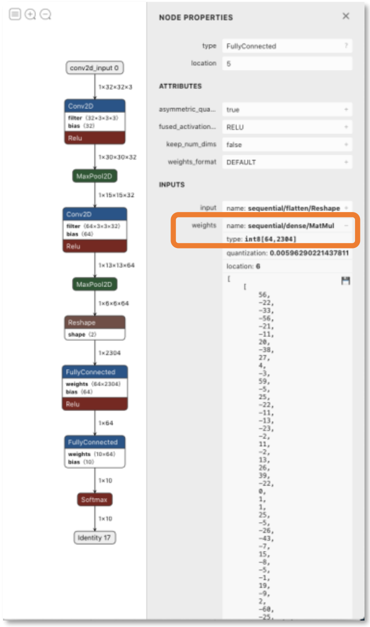
weights: name: **sequential/dense/MatMul**

type: **float32[64,2304]**

location: **6**

```
[
    [
        0.3339117765426636,
        -0.12951909005641937
        -0.19890105724334717
        -0.3362927734851837,
        -0.12596379220485687
        -0.06459484994411469
        0.1176003119707108,
        -0.22865332663059235
        0.16351580619812012,
        0.02213565818965435,
        -0.02005589194595813
        0.3515489101409912,
        -0.03274595737457275
        0.15197747945785522,
        -0.1288449466228485,
        -0.06321356445550919
        -0.07943038642406464
        -0.13694220781326294
        -0.00915299635380506
        0.06841003894805908,
        -0.01234686467796564
        0.07489526271820068,
        0.15777553617954254,
        0.23148055374622345,
        -0.12965452671051025
        0.002687103347852826
        0.008302353322505951
        0.003798437770456075
        0.146267369389534,
        -0.02969614788889885
        -0.15224052965641022
        -0.2562084496021271,
        -0.04458019509911537
        0.08812304586172104,
        -0.04942338168621063
        -0.03026680275797844
        -0.00539603084325790
```

Cifar_10.tflite



NODE PROPERTIES

type: FullyConnected

location: 5

ATTRIBUTES

asymmetric_qua...: true

fused_activation...: RELU

keep_num_dims: false

weights_format: DEFAULT

INPUTS

input: name: **sequential/flatten/Reshape**

weights: name: **sequential/dense/MatMul**

type: **int8[64,2304]**

quantization: **0.00596290221437811**

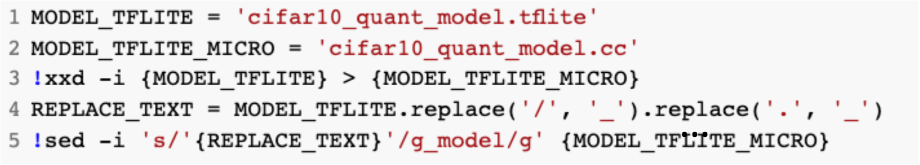
location: **6**

```
[
    [
        56,
        -22,
        -33,
        -56,
        -21,
        -11,
        20,
        -38,
        27,
        4,
        -3,
        59,
        -5,
        25,
        -22,
        -11,
        -13,
        -23,
        -2,
        11,
        -2,
        13,
        26,
        39,
        -22,
        0,
        1,
        1,
        25,
        -5,
        -26,
        -43,
        -7,
        15,
        -8,
        -5,
        -1,
        19,
        -9,
        2,
        -60,
        -25,
```

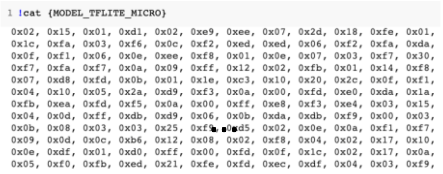
Cifar_quant_10.tflite

Generate a TF Lite for Micro Model

To convert the TensorFlow Lite quantized model into a C source file that can be loaded by TensorFlow Lite for Microcontrollers on MCUs - use Linux **xxd** tool to convert the .tflite file into a .cc file.

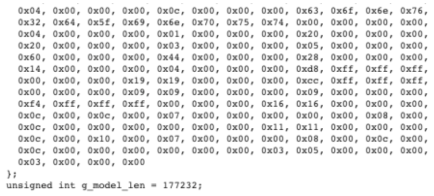


```
MODEL_TFLITE = 'cifar10_quant_model.tflite'
MODEL_TFLITE_MICRO = 'cifar10_quant_model.cc'
!xxd -i {MODEL_TFLITE} > {MODEL_TFLITE_MICRO}
REPLACE_TEXT = MODEL_TFLITE.replace('/', '_').replace('.', '_')
!sed -i 's/'{REPLACE_TEXT}'/g_model/g' {MODEL_TFLITE_MICRO}
```



```
!cat {MODEL_TFLITE_MICRO}
```

```
0x02, 0x15, 0x01, 0xd1, 0x02, 0xe9, 0xee, 0x07, 0x2d, 0x18, 0xfe, 0x01,
0x1c, 0xfa, 0x03, 0xf6, 0x0c, 0xf2, 0xed, 0xed, 0x06, 0xf2, 0xfa, 0xda,
0x0f, 0xf1, 0x06, 0x0e, 0xee, 0xf8, 0x01, 0x0e, 0x07, 0x03, 0xf7, 0x30,
0xf7, 0xfa, 0xf7, 0x0a, 0x09, 0xff, 0x12, 0x02, 0xfb, 0x01, 0x14, 0xf8,
0x07, 0xd8, 0xfd, 0x0b, 0x01, 0x1e, 0xc3, 0x10, 0x20, 0x2c, 0x0f, 0xf1,
0x04, 0x10, 0x05, 0x2a, 0xd9, 0xf3, 0x0a, 0x00, 0xfd, 0xe0, 0xda, 0x1a,
0xfb, 0xea, 0xfd, 0xf5, 0x0a, 0x00, 0xff, 0xe8, 0xf3, 0xe4, 0x03, 0x15,
0x04, 0x0d, 0xff, 0xdb, 0xd9, 0x06, 0x0b, 0xda, 0xdb, 0xf9, 0x00, 0x03,
0x0b, 0x08, 0x03, 0x03, 0x25, 0xf9, 0xd5, 0x02, 0x0e, 0x0a, 0xf1, 0xf7,
0x09, 0x0d, 0x0c, 0xb6, 0x12, 0x08, 0x02, 0xf8, 0x04, 0x02, 0x17, 0x10,
0x0e, 0xdf, 0x01, 0xd0, 0xff, 0x00, 0xfd, 0x0f, 0x1c, 0x02, 0x17, 0x0a,
0x05, 0xf0, 0xfb, 0xed, 0x21, 0xfe, 0xfd, 0xec, 0xdf, 0x04, 0x03, 0xf9,
```



```
  0x04, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x63, 0x6f, 0x6e, 0x76,
  0x32, 0x64, 0x5f, 0x69, 0x6e, 0x70, 0x75, 0x74, 0x00, 0x00, 0x00, 0x00,
  0x04, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00,
  0x20, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00,
  0x60, 0x00, 0x00, 0x00, 0x44, 0x00, 0x00, 0x00, 0x28, 0x00, 0x00, 0x00,
  0x14, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0xd8, 0xff, 0xff, 0xff,
  0x00, 0x00, 0x00, 0x19, 0x19, 0x00, 0x00, 0x00, 0xcc, 0xff, 0xff, 0xff,
  0x00, 0x00, 0x00, 0x09, 0x09, 0x00, 0x00, 0x00, 0x09, 0x00, 0x00, 0x00,
  0xf4, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x16, 0x16, 0x00, 0x00, 0x00,
  0x0c, 0x00, 0x0c, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x00,
  0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x11, 0x11, 0x00, 0x00, 0x00,
  0x0c, 0x00, 0x10, 0x00, 0x07, 0x00, 0x00, 0x00, 0x08, 0x00, 0x0c, 0x00,
  0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x05, 0x00, 0x00, 0x00,
  0x03, 0x00, 0x00, 0x00
};
unsigned int g_model_len = 177232;
```

Image Classification (Inference) Using TF-Lite

(Reloaded) Code Time!

CNN_Cifar_10_TFLite.ipynb

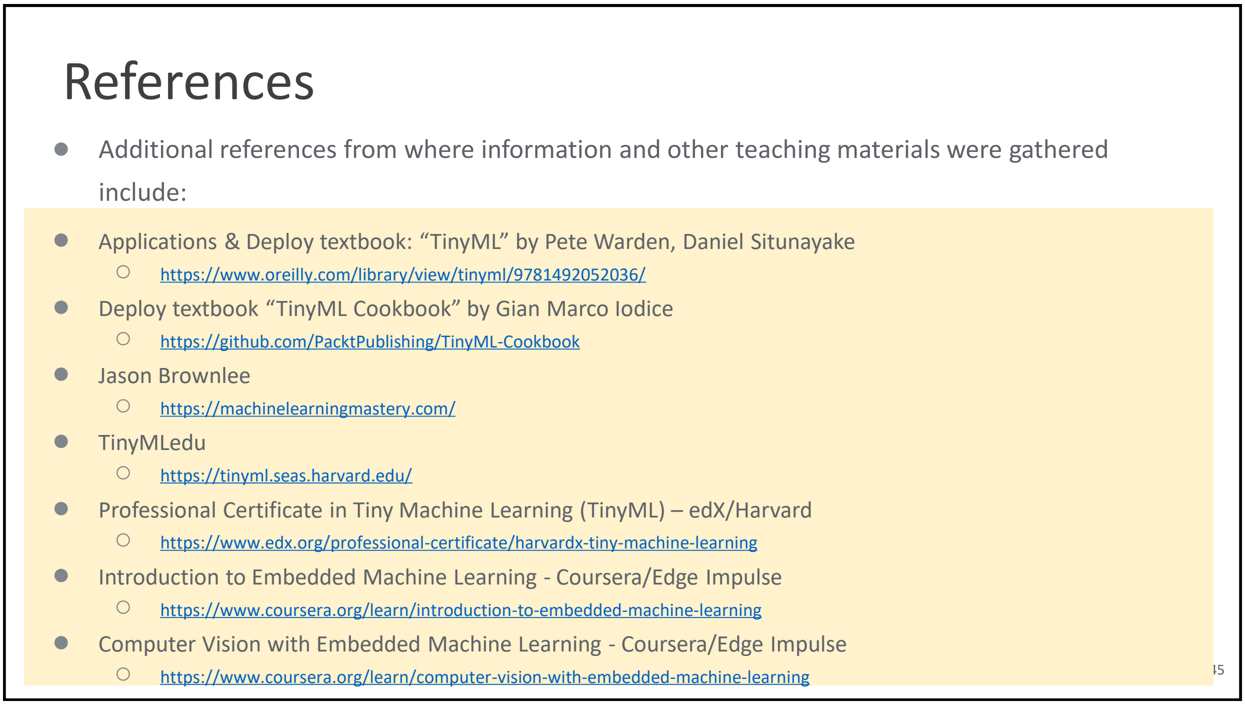
TFLite Micro: “Hello World”

Code Time!

train_TFL_Micro_hello_world_model.ipynb

Credits

- A previous edition of this course was developed in collaboration with Dr. Susan C. Schneider of Marquette University.
- We are very grateful and thank all the following professors, researchers, and practitioners for jump-starting courses on TinyML and for sharing their teaching materials:
- Prof. Marcelo Rovai - TinyML - Machine Learning for Embedding Devices, UNIFEI
  - https://github.com/Mjrovai/UNIFEI-IESTI01-TinyML-2022.1
- Prof. Vijay Janapa Reddi - CS249r: Tiny Machine Learning, Applied Machine Learning on Embedded IoT Devices, Harvard
  - https://sites.google.com/g.harvard.edu/tinyml/home
- Prof. Rahul Mangharam – ESE3600: Tiny Machine Learning, Univ. of Pennsylvania
  - https://tinyml.seas.upenn.edu/#
- Prof. Brian Plancher - Harvard CS249r: Tiny Machine Learning (TinyML), Barnard College, Columbia University
  - https://a2r-lab.org/courses/cs249r_tinyml/



References

- Additional references from where information and other teaching materials were gathered include:
- Applications & Deploy textbook: “TinyML” by Pete Warden, Daniel Situnayake
  - ○ https://www.oreilly.com/library/view/tinyml/9781492052036/
- Deploy textbook “TinyML Cookbook” by Gian Marco Iodice
  - ○ https://github.com/PacktPublishing/TinyML-Cookbook
- Jason Brownlee
  - ○ https://machinelearningmastery.com/
- TinyMLedu
  - ○ https://tinyml.seas.harvard.edu/
- Professional Certificate in Tiny Machine Learning (TinyML) – edX/Harvard
  - ○ https://www.edx.org/professional-certificate/harvardx-tiny-machine-learning
- Introduction to Embedded Machine Learning - Coursera/Edge Impulse
  - ○ https://www.coursera.org/learn/introduction-to-embedded-machine-learning
- Computer Vision with Embedded Machine Learning - Coursera/Edge Impulse
  - ○ https://www.coursera.org/learn/computer-vision-with-embedded-machine-learning