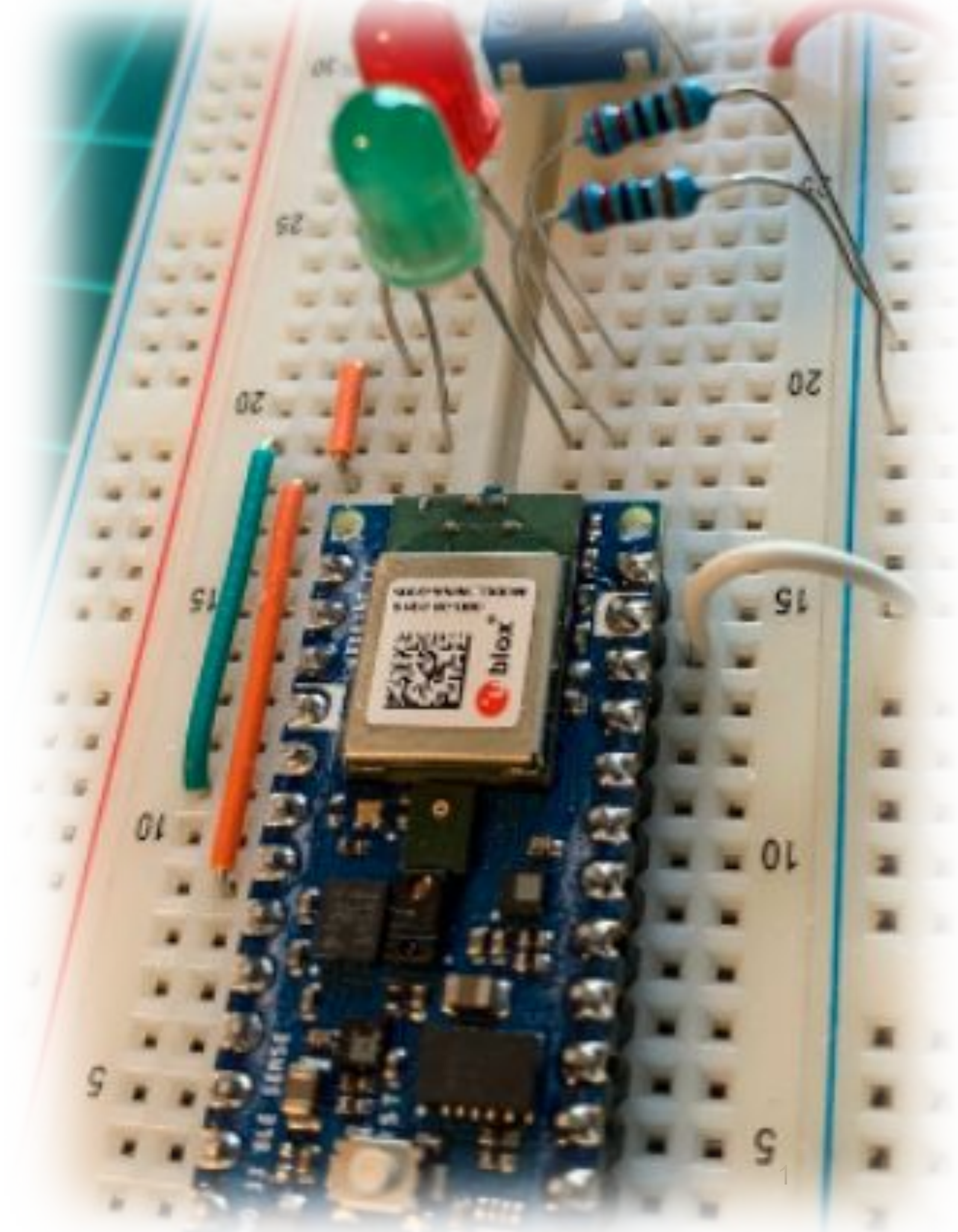
IESTI01 – TinyML

Embedded Machine Learning

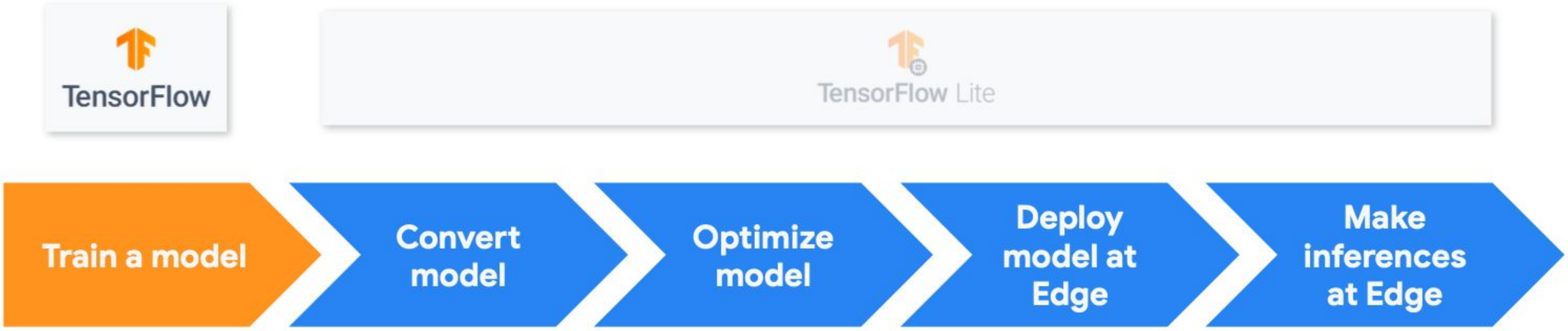
16. Introduction to TensorFlow Lite and TFL-Micro

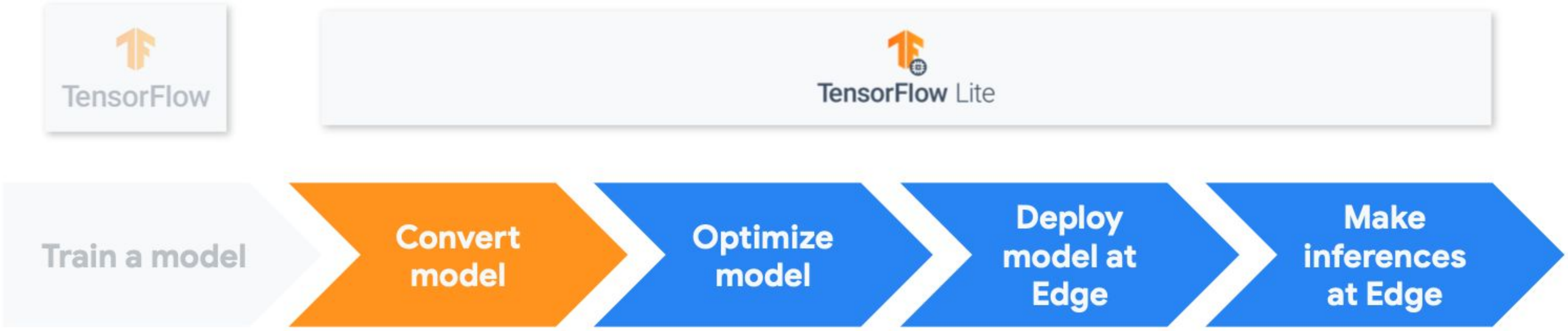
Prof. Marcelo Rovai
UNIFEI

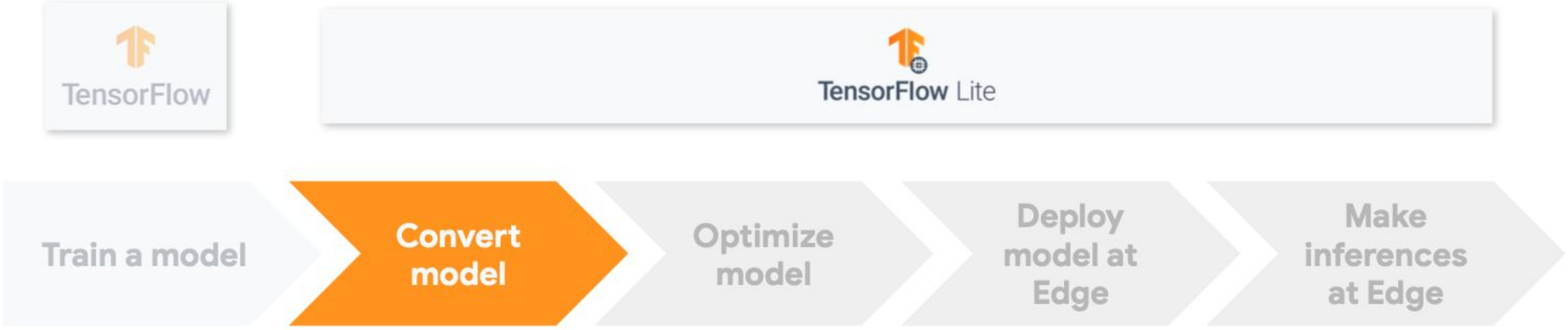


Introduction to TFLite

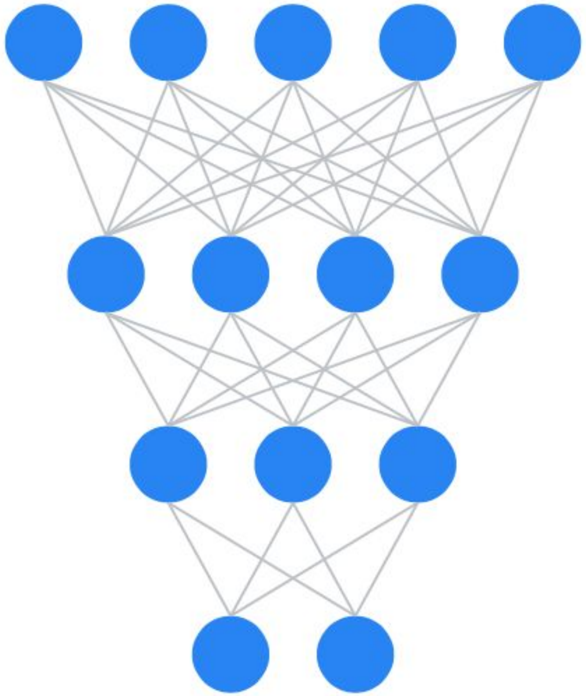
Inference at the Edge


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

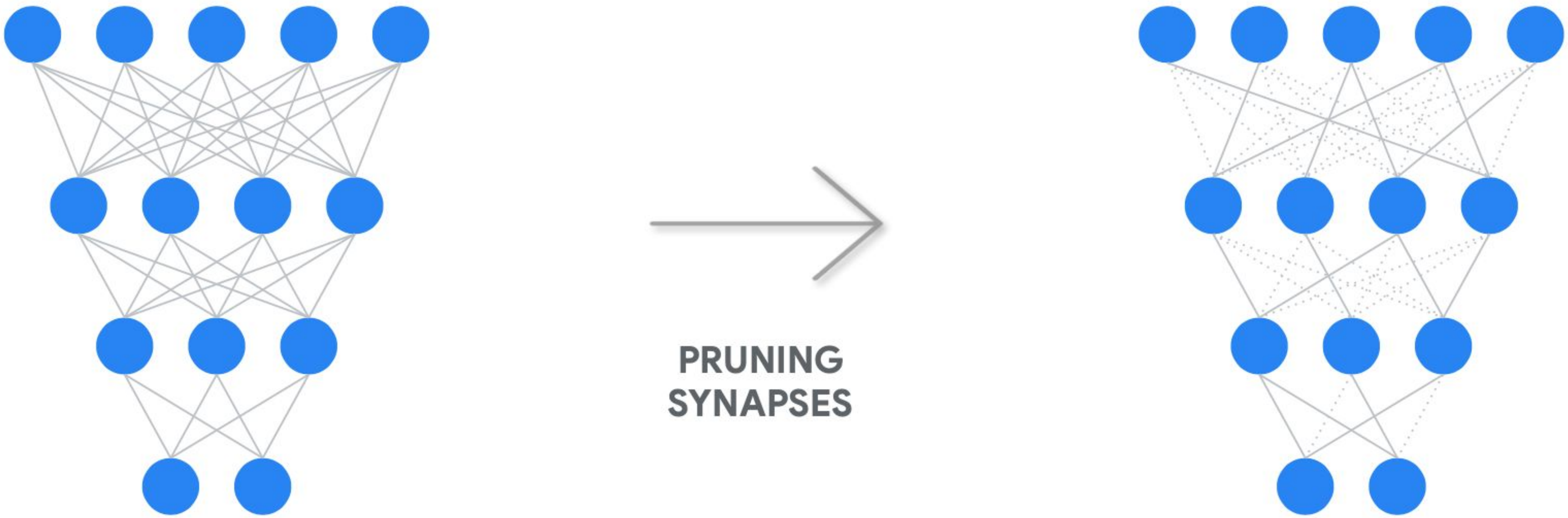

TensorFlow
TensorFlow Lite
Train a model
Convert model
Optimize model
Deploy model at Edge
Make inferences at Edge


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

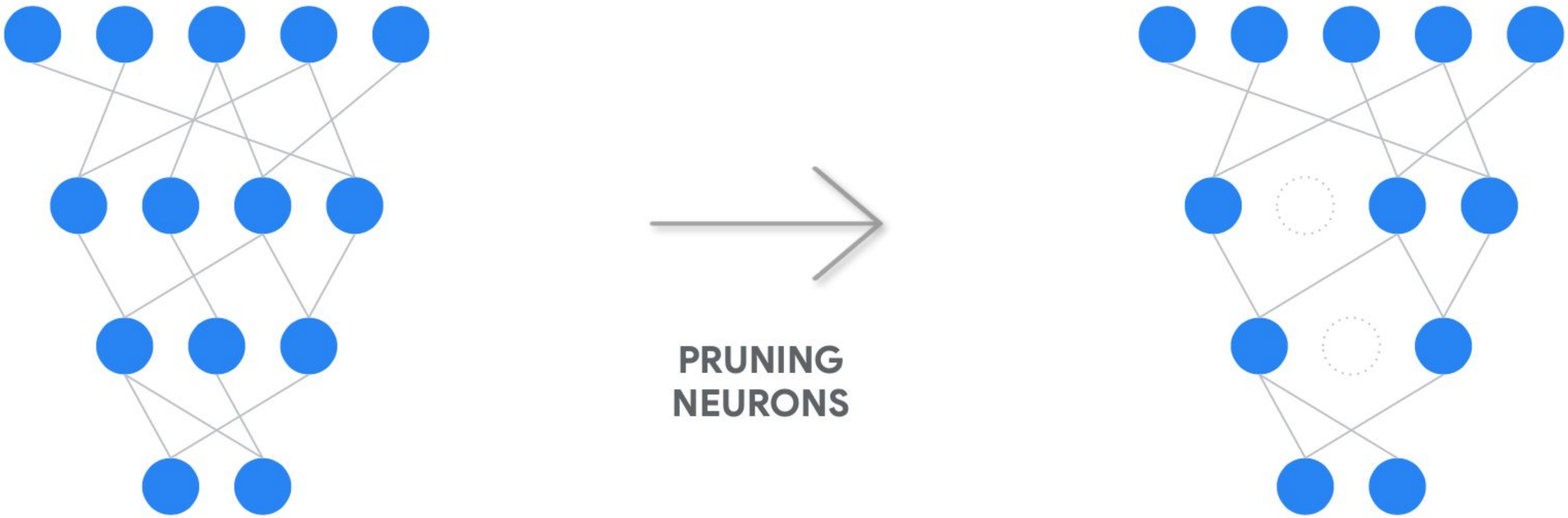
Pruning



Pruning



Pruning



More info: An introduction to weight pruning by Tivadar Danka

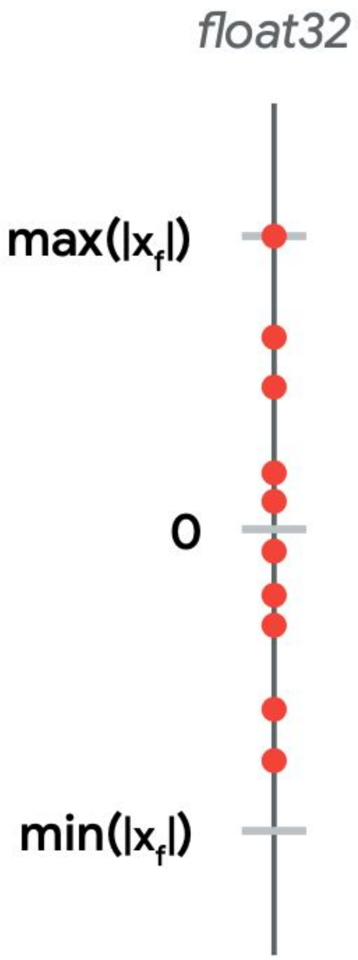
Quantization

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:

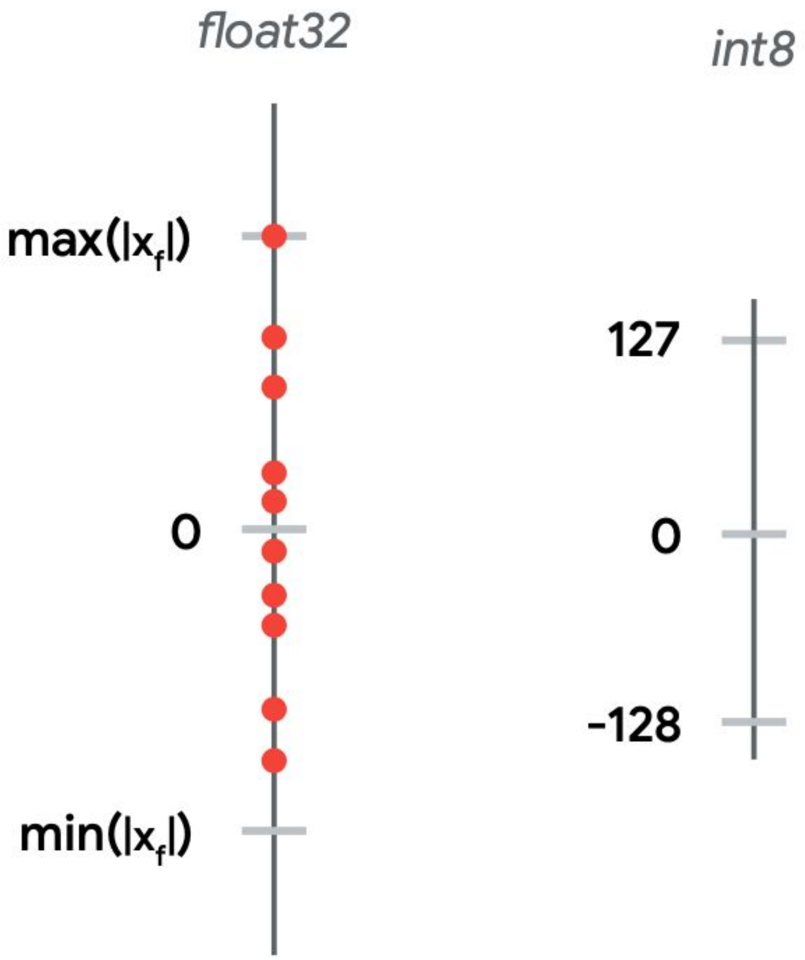
✔ **smaller model size**,
✔ **better portability (*)** and
✔ **faster computation**.

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

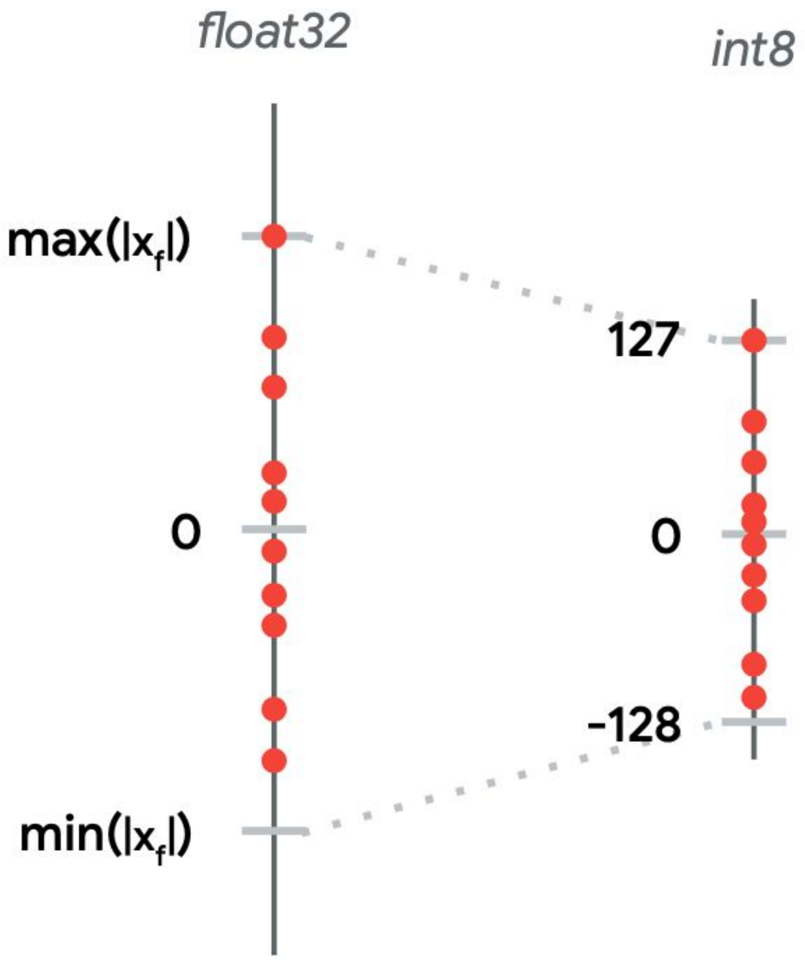
Quantization



Quantization



Quantization



	Floating-point Baseline
MobileNet v1 1.0 224	71.03%
MobileNet v2 1.0 224	70.77%
Resnet v1 50	76.30%

	Floating-point Baseline	Post-training Quantization (PTQ)*
MobileNet v1 1.0 224	71.03%	69.57%
MobileNet v2 1.0 224	70.77%	70.20%
Resnet v1 50	76.30%	75.95%

PTQ is the most common method of quantization. You can also apply QAT (Quantization-aware training), where the parameters are quantized **during** training.

	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%

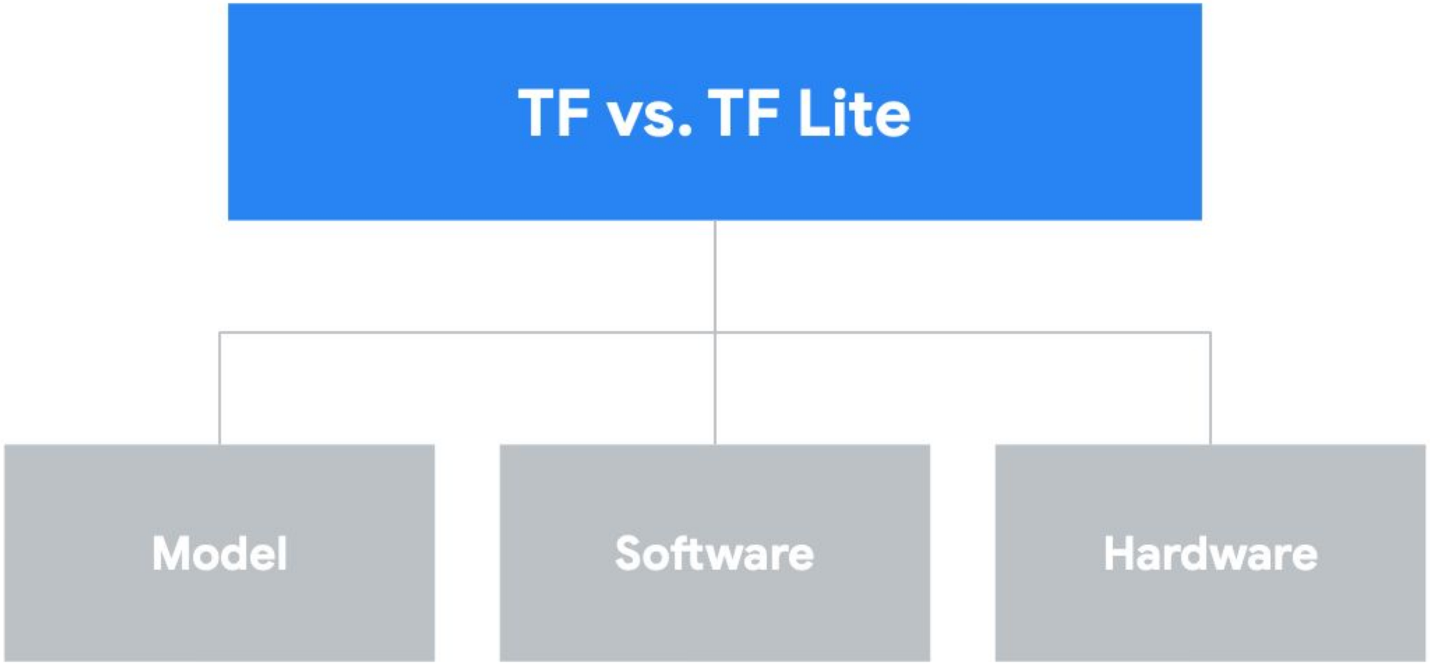
More info: How to accelerate and compress neural networks with quantization

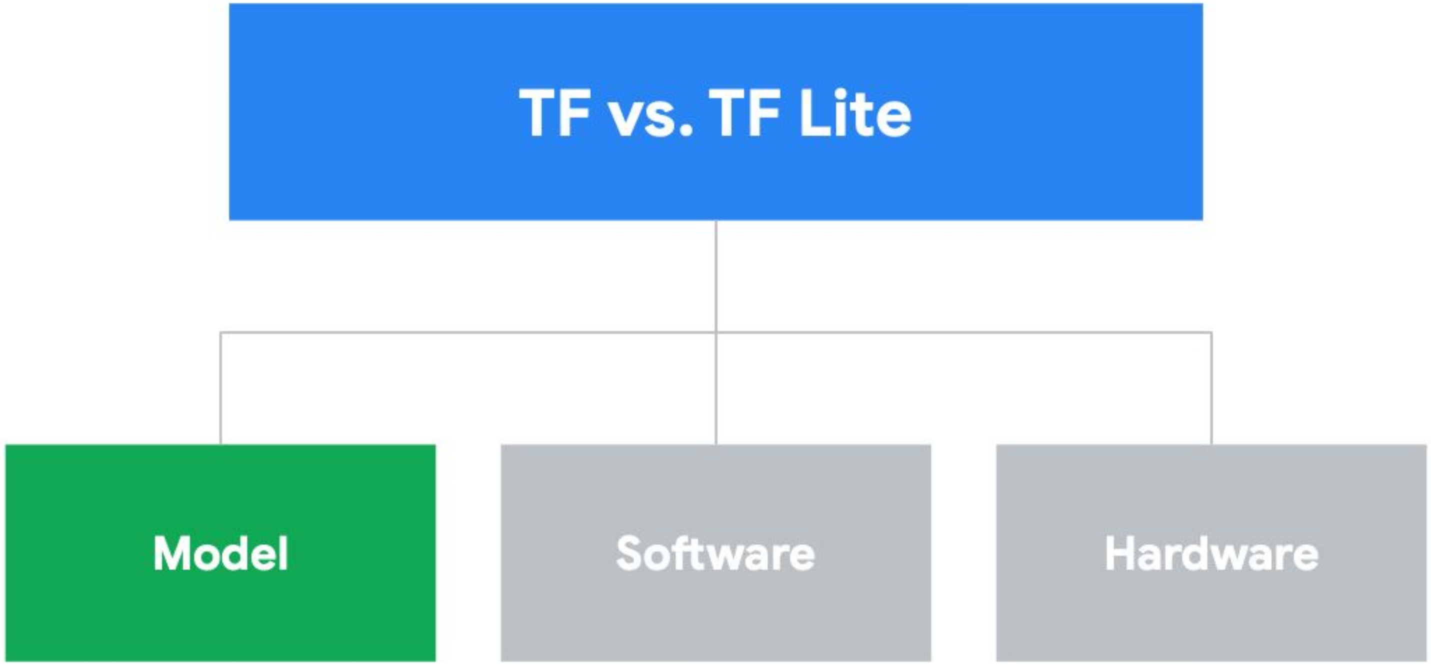
TensorFlow

TensorFlow Lite

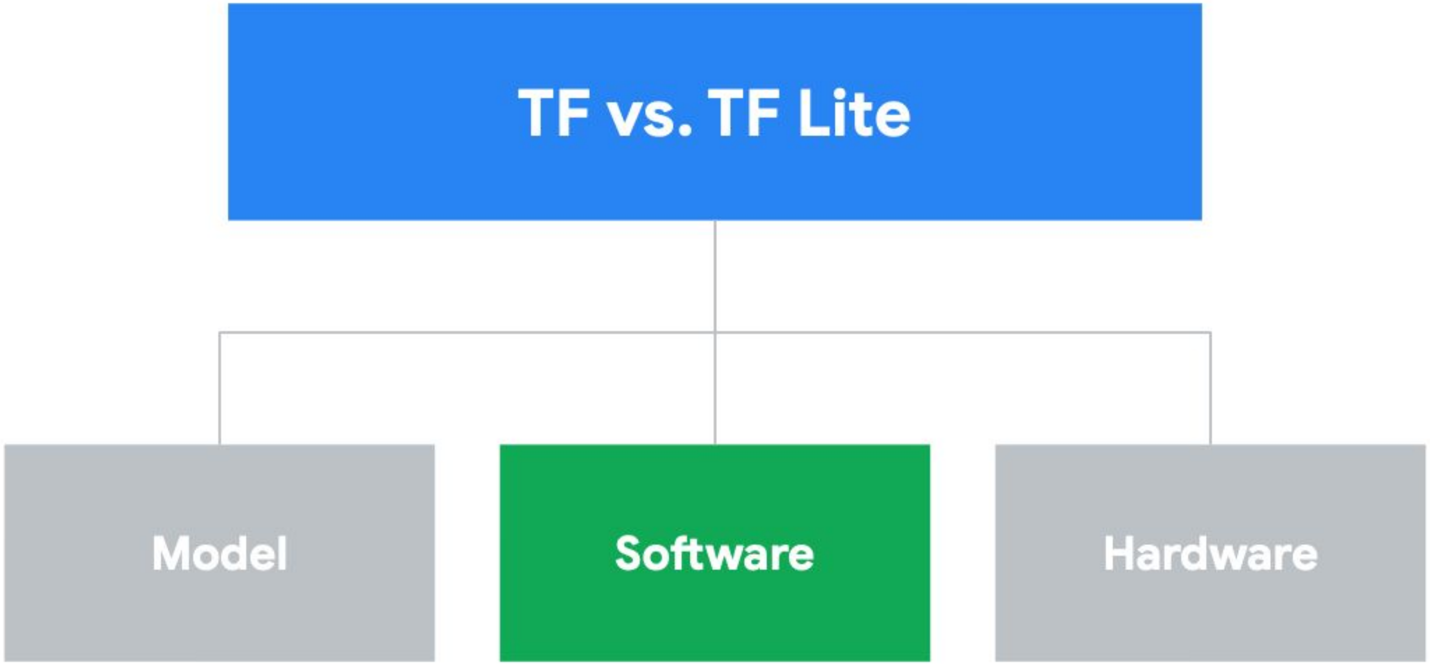
Key Differences

	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

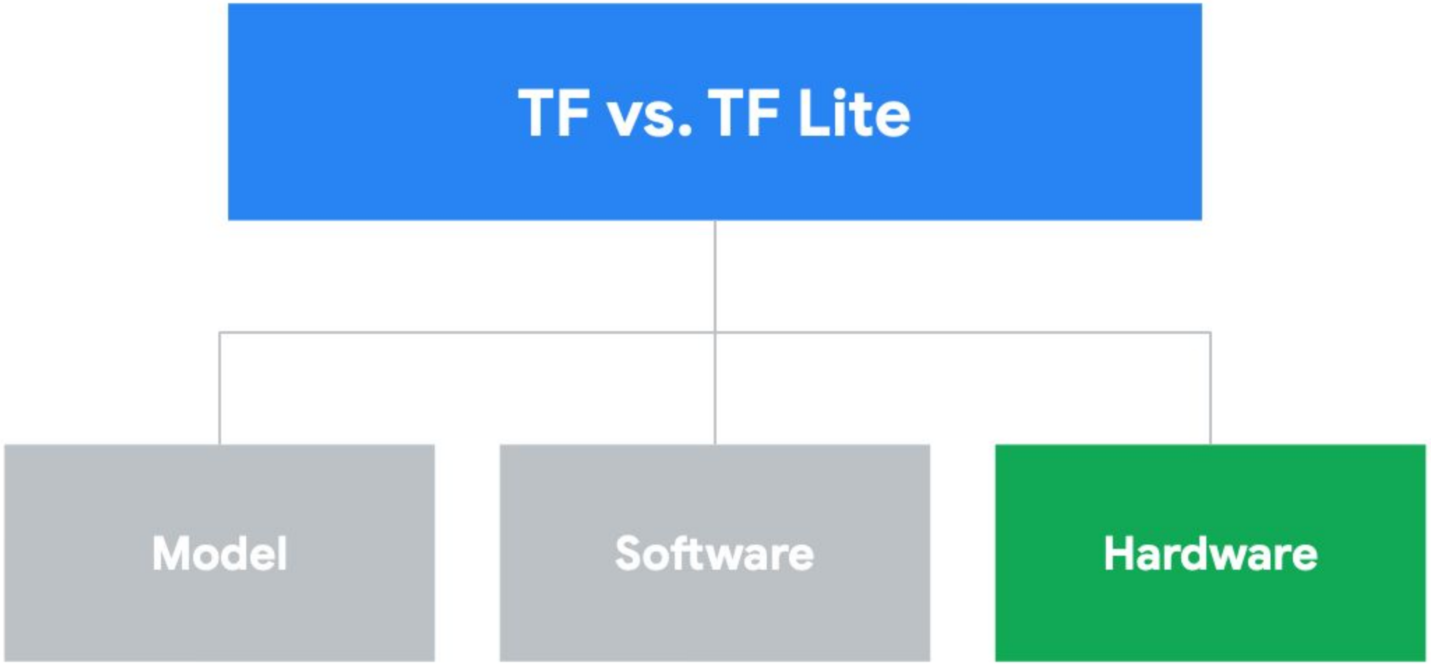

TF vs. TF Lite
Model
Software
Hardware


TF vs. TF Lite
Model
Software
Hardware

	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


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

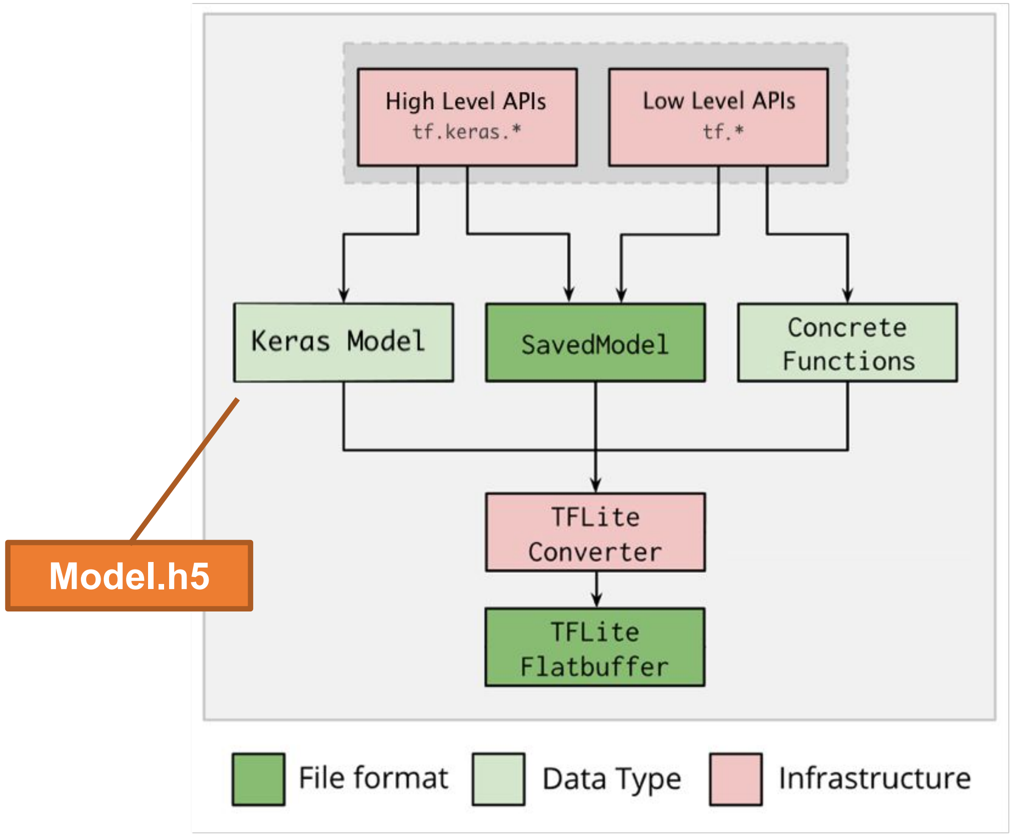

TF vs. TF Lite
Model
Software
Hardware

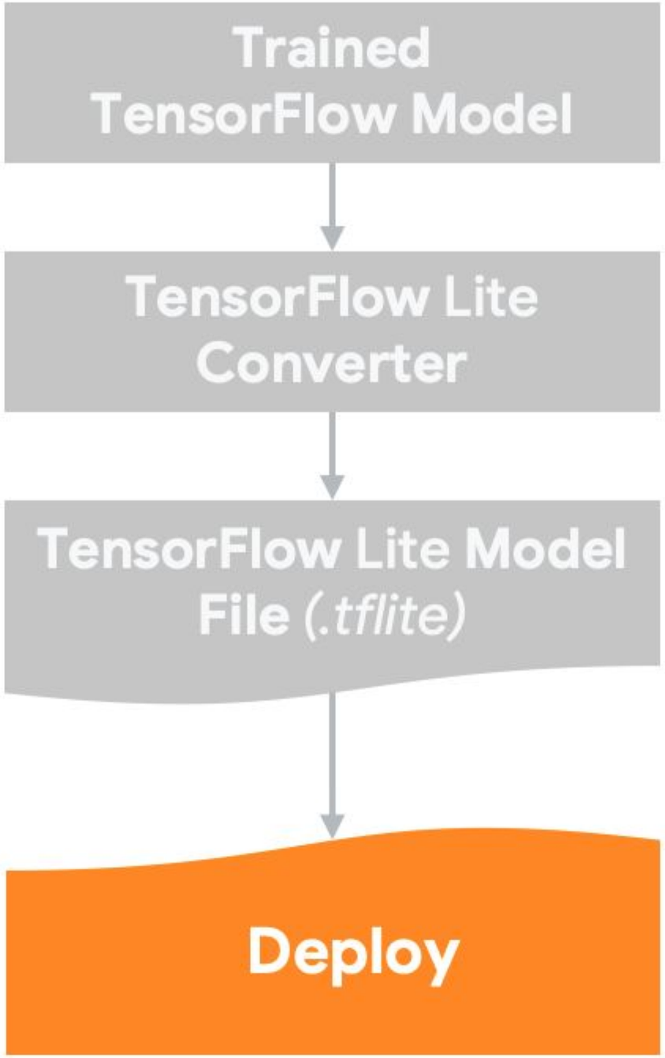
	TensorFlow	TensorFlow Lite	TensorFlow Lite Micro
Base Binary Size	3MB+	100KB	~10 KB
Base Memory Footprint	~5MB	300KB	20KB
Optimized Architectures	X86, TPUs, GPUs	Arm Cortex A, x86	Arm Cortex M, DSPs, MCUs

Optimization and Quantization

Minimizing compression loss

TensorFlow **Workflow**





Converting

Size: 2.1Mb

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

TF Model

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

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

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

TFLite Model

```
673324
```

Size: .63Mb

Converting from a saved model

Size: 2.1Mb

```
[81]   1 model_path = '/content/cifar_10_model.h5'
```

TF Model

```
[82]   1 model_cifar10 = tf.keras.models.load_model(model_path)
```

```
[83]   1 converter = tf.lite.TFLiteConverter.from_keras_model(model_cifar10)
```

```
[84]   1 tflite_model = converter.convert()

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

Save tflite model

Size: .63Mb

```
[85]   1 open("/content/cifar10.tflite","wb").write(tflite_model)

673324
```

TFLite Model

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()

```

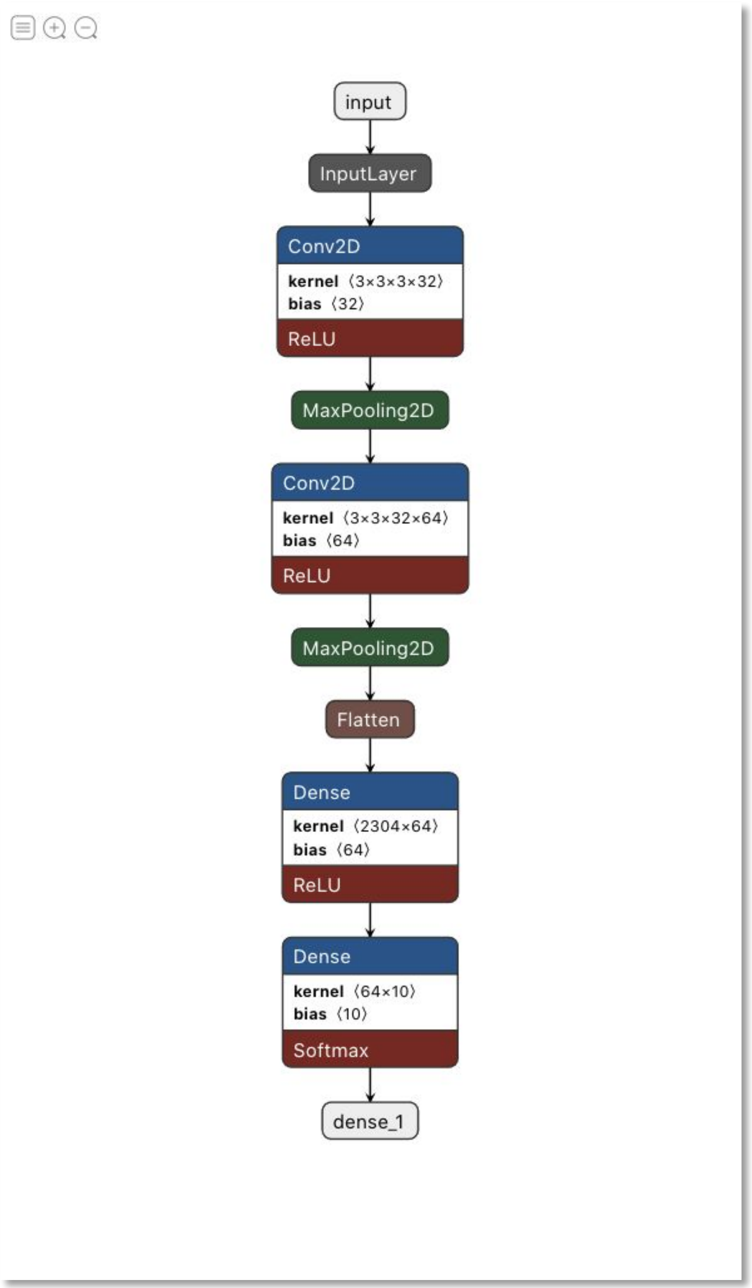
```
INFO:tensorflow:Assets written to: /tmp/tmpyyiq46sj/assets
INFO:tensorflow:Assets written to: /tmp/tmpyyiq46sj/assets
```

Size: .18Mb

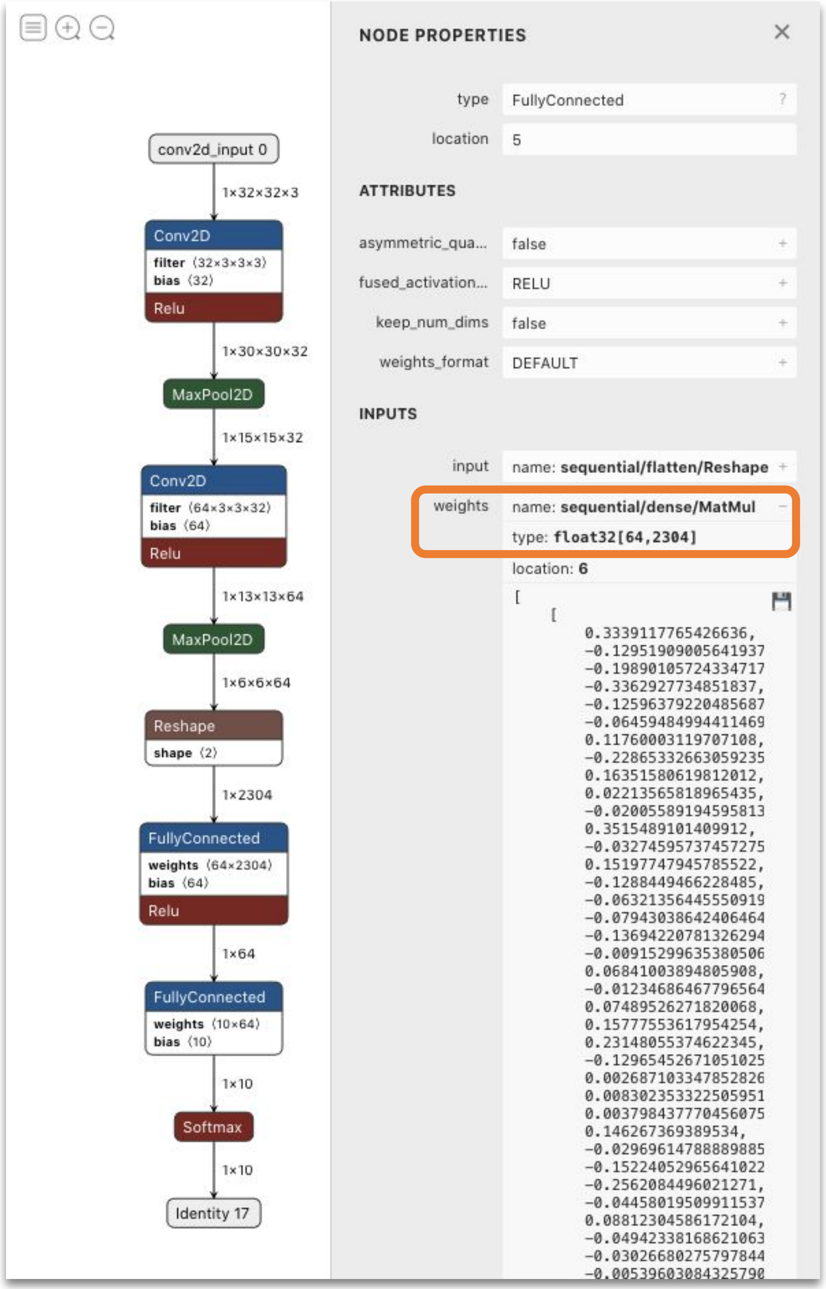
[75]

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

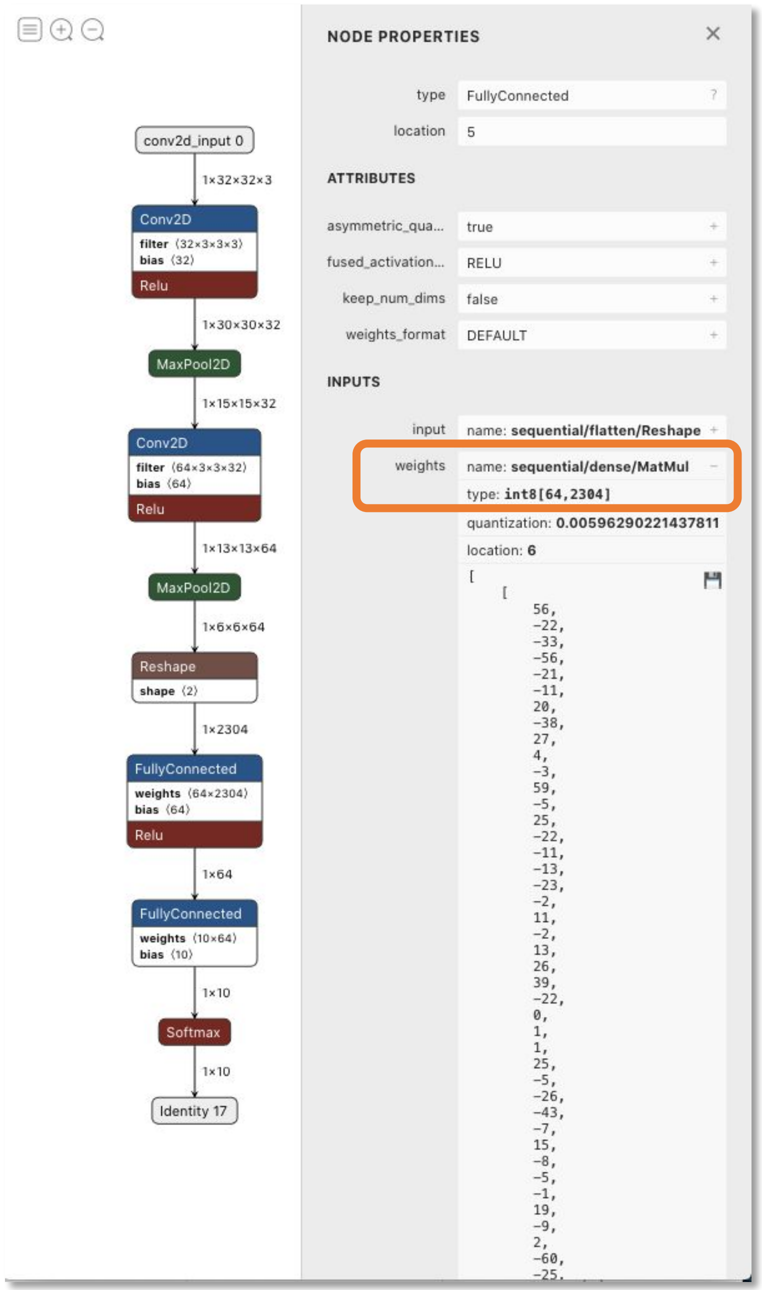
```
177232
```



Cifar_10.h5



Cifar_10.tflite



Cifar_quant_10.tflite

Generating 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, we simply need to use the 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

Code Time!

CNN_Cifar_10_TFLite.ipynb

TFLite-Micro: “Hello World”

Code Time!

train_TFL_Micro_hello_world_model.ipynb

Reading Material

Main references

- Harvard School of Engineering and Applied Sciences - CS249r: Tiny Machine Learning
- Professional Certificate in Tiny Machine Learning (TinyML) – edX/Harvard
- Introduction to Embedded Machine Learning - Coursera/Edge Impulse
- Computer Vision with Embedded Machine Learning - Coursera/Edge Impulse
- Fundamentals textbook: "Deep Learning with Python" by François Chollet
- Applications & Deploy textbook: "TinyML" by Pete Warden, Daniel Situnayake
- Deploy textbook "TinyML Cookbook" by Gian Marco Iodice

I want to thank Shawn Hymel and Edge Impulse, Pete Warden and Laurence Moroney from Google, Professor Vijay Janapa Reddi and Brian Plancher from Harvard, and the rest of the TinyMLedu team for preparing the excellent material on TinyML that is the basis of this course at UNIFEI.

The IESTI01 course is part of the **TinyML4D**, an initiative to make TinyML education available to everyone globally.

Thanks

UNIFEI