

# IESTI01 – TinyML

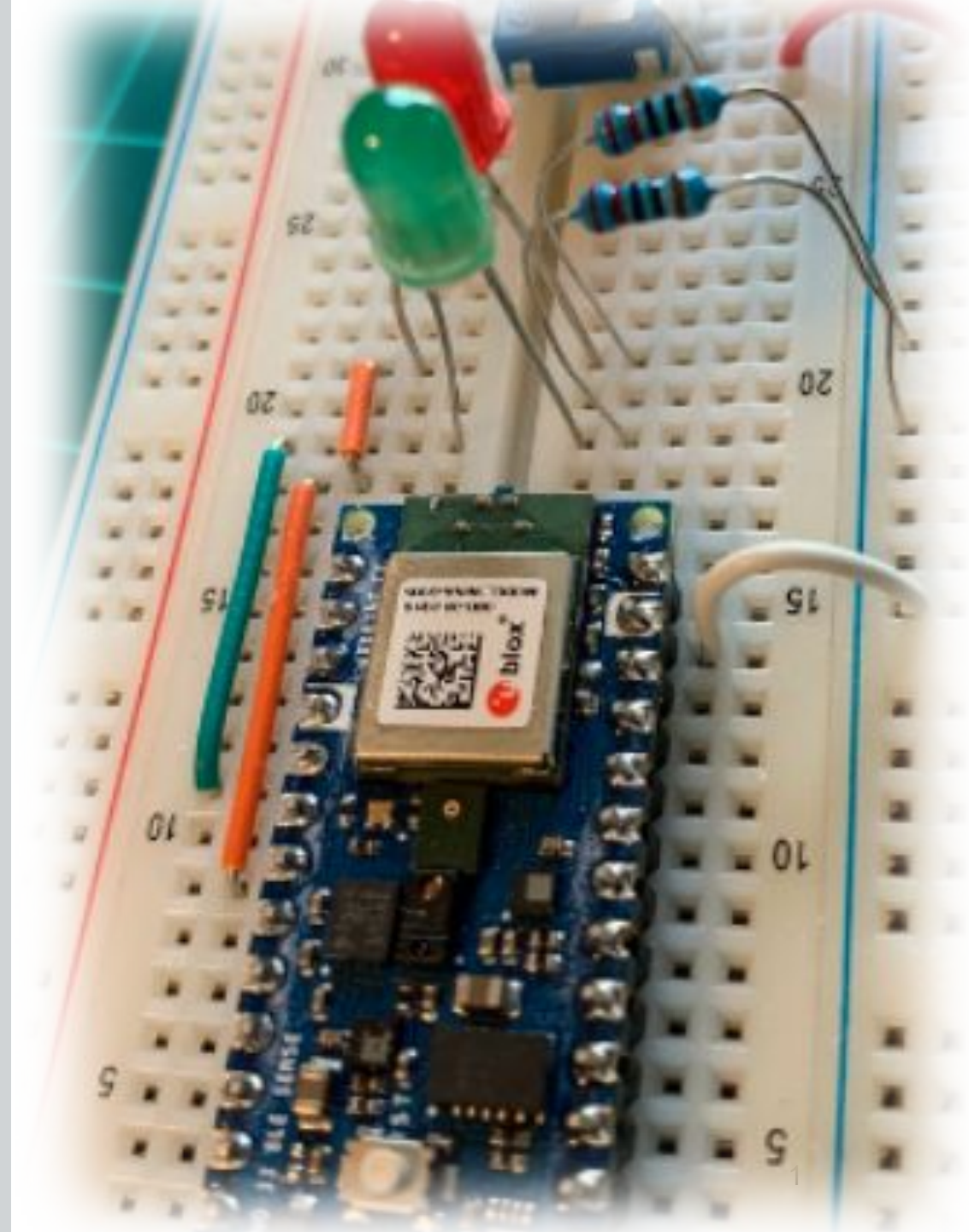
## Embedded Machine Learning

### 16. Introduction to TensorFlow Lite and TFL-Micro



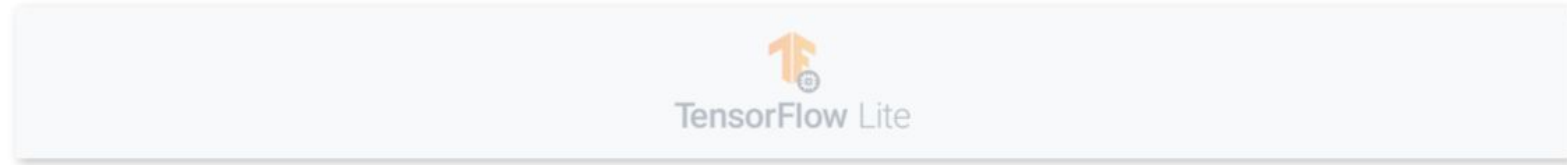
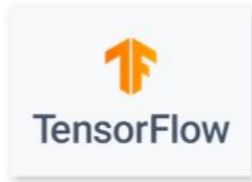
Prof. Marcelo Rovai

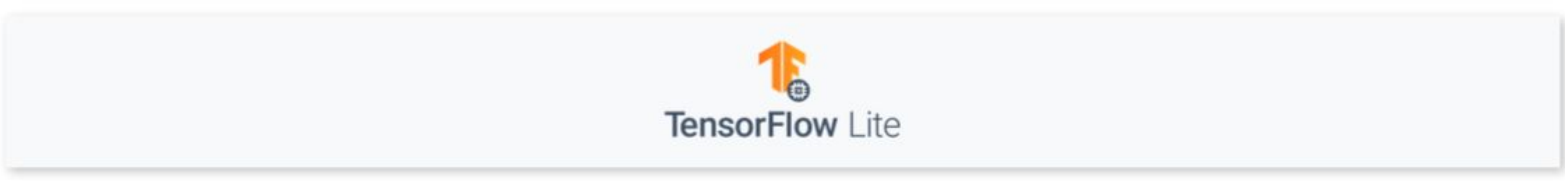
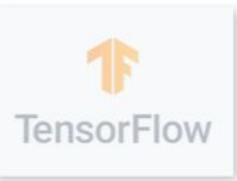
UNIFEI

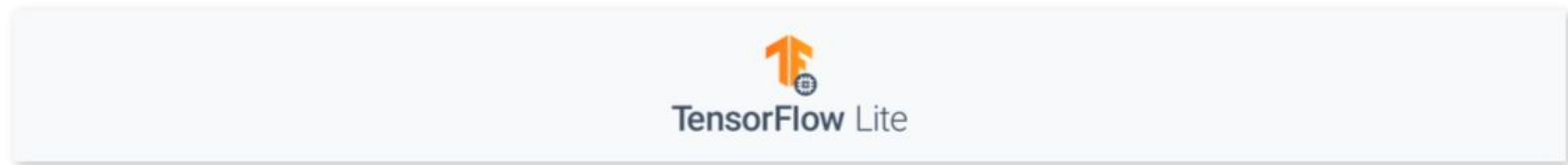
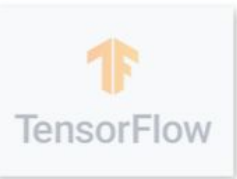


# Introduction to TFLite

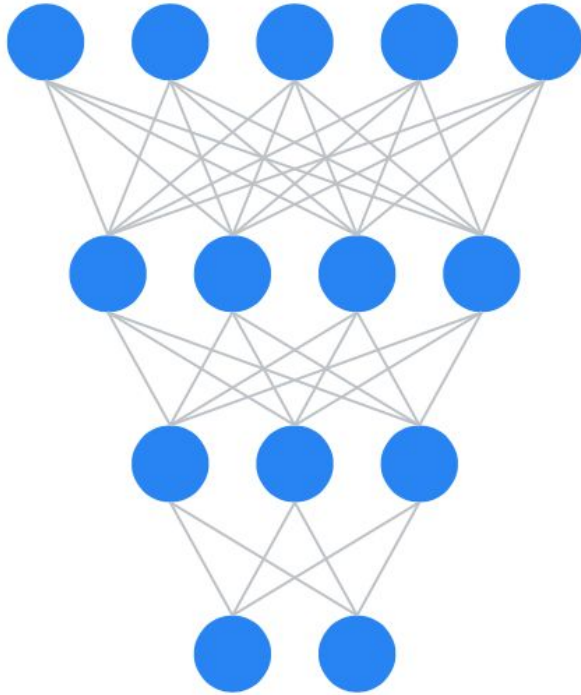
## Inference at the Edge



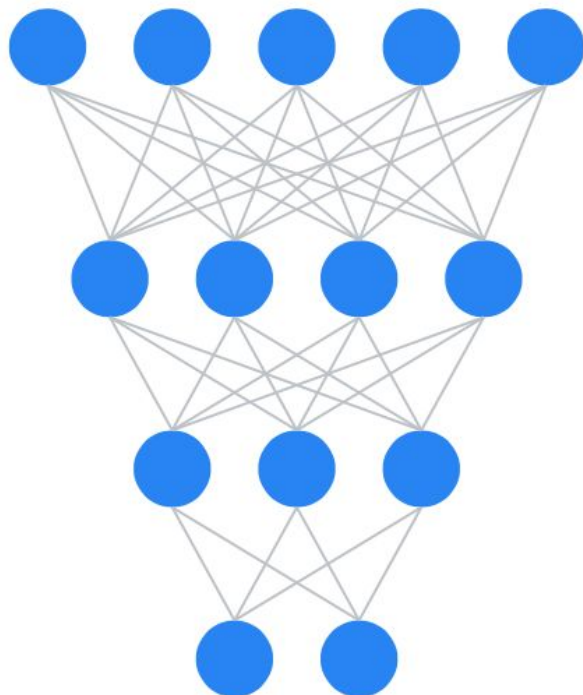




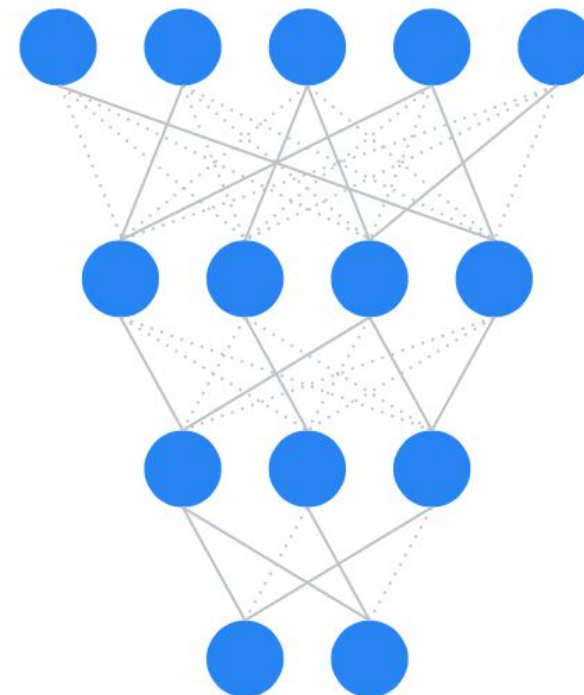
# Pruning



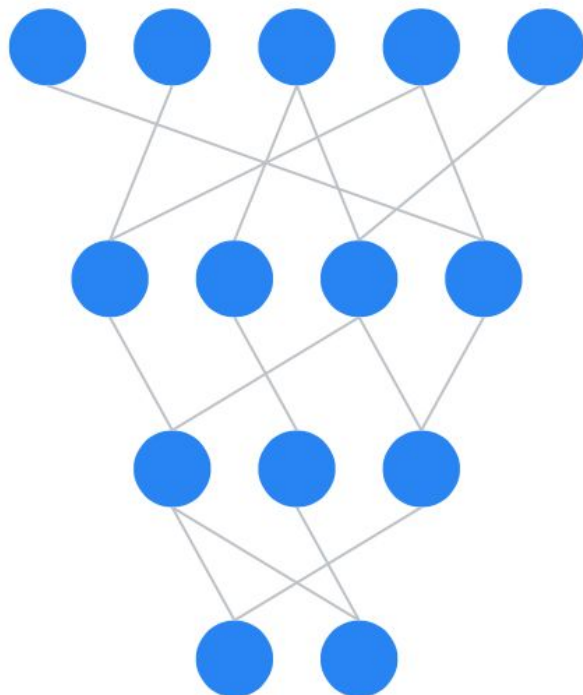
# Pruning



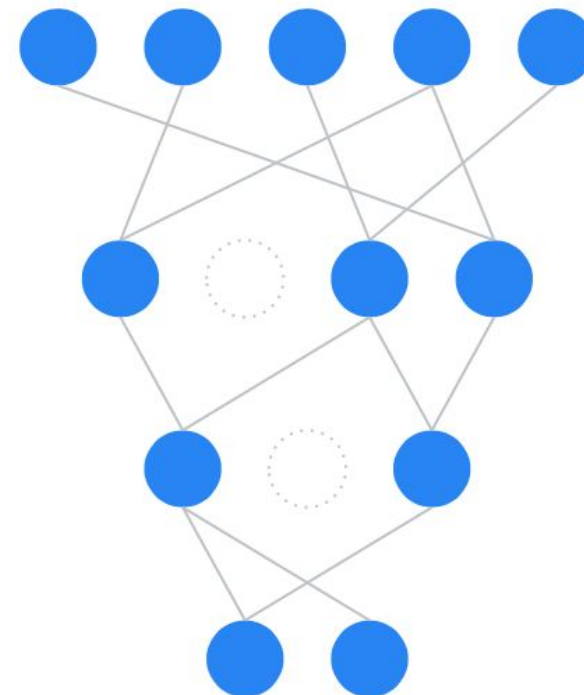
**PRUNING  
SYNAPSES**



# Pruning



**PRUNING  
NEURONS**





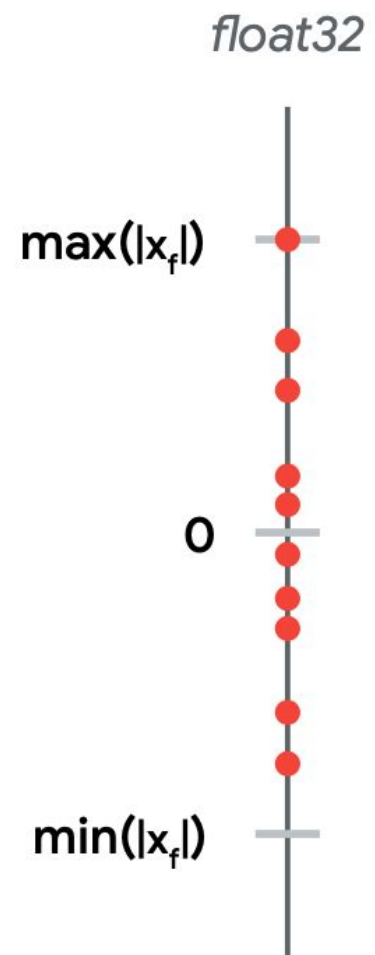
# Quantization

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:

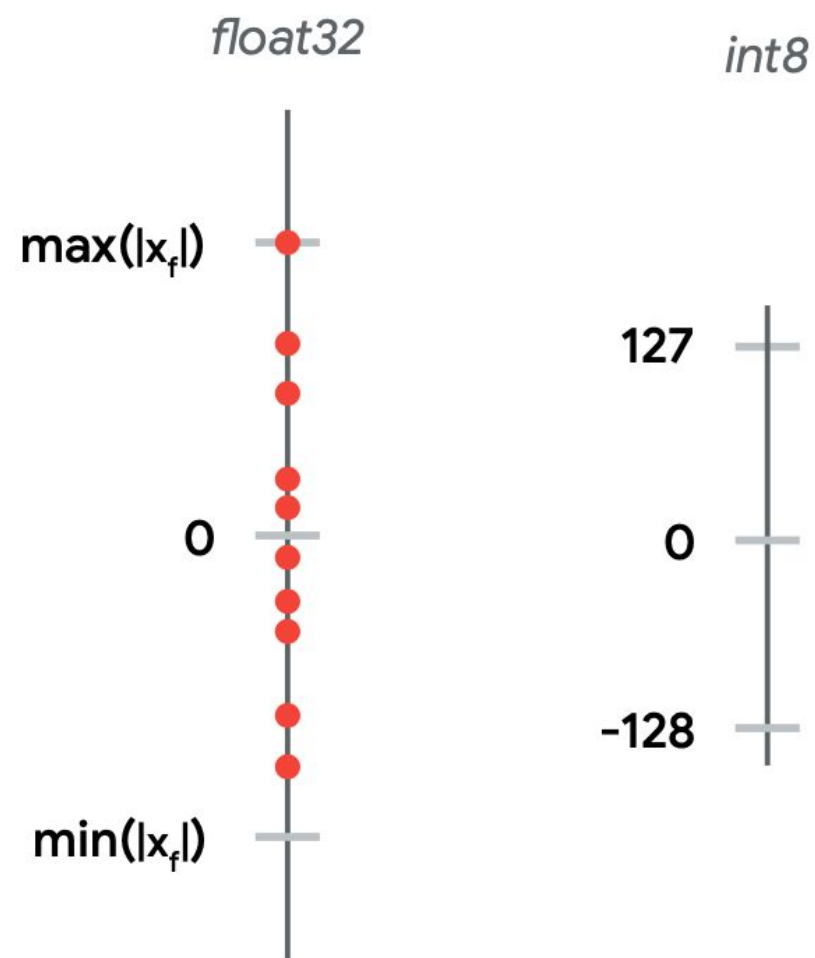
- ✓ **smaller model size,**
- ✓ **better portability (\*)** and
- ✓ **faster computation.**

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

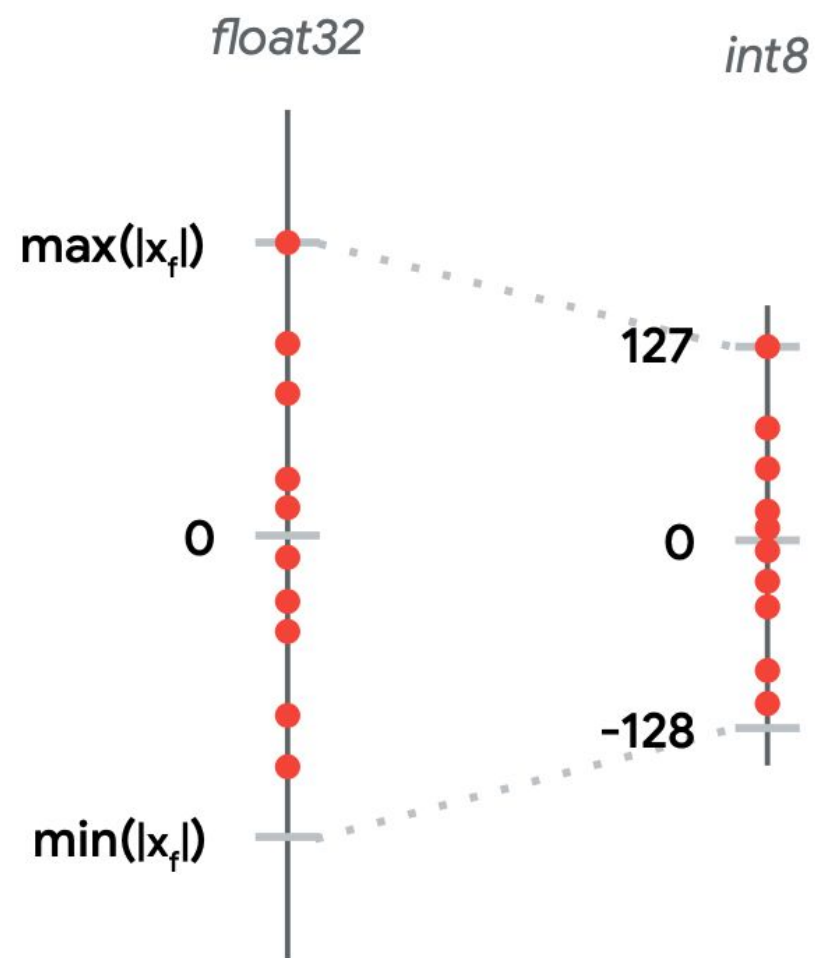
# Quantization



# 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%



TensorFlow



TensorFlow Lite



# Key Differences

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

## TF vs. TF Lite

```
graph TD; A[TF vs. TF Lite] --> B[Model]; A --> C[Software]; A --> D[Hardware];
```

**Model**

**Software**

**Hardware**

## TF vs. TF Lite

```
graph TD; A[TF vs. TF Lite] --> B[Model]; A --> C[Software]; A --> D[Hardware];
```

**Model**

**Software**

**Hardware**



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

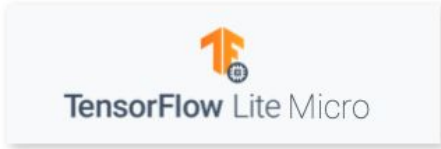
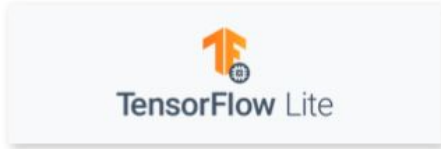
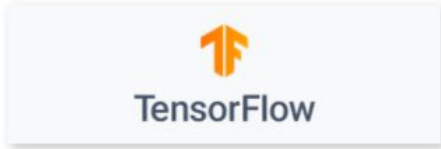
## TF vs. TF Lite

```
graph TD; A[TF vs. TF Lite] --> B[Model]; A --> C[Software]; A --> D[Hardware];
```

Model

Software

Hardware



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

## TF vs. TF Lite

```
graph TD; A[TF vs. TF Lite] --> B[Model]; A --> C[Software]; A --> D[Hardware];
```

**Model**

**Software**

**Hardware**



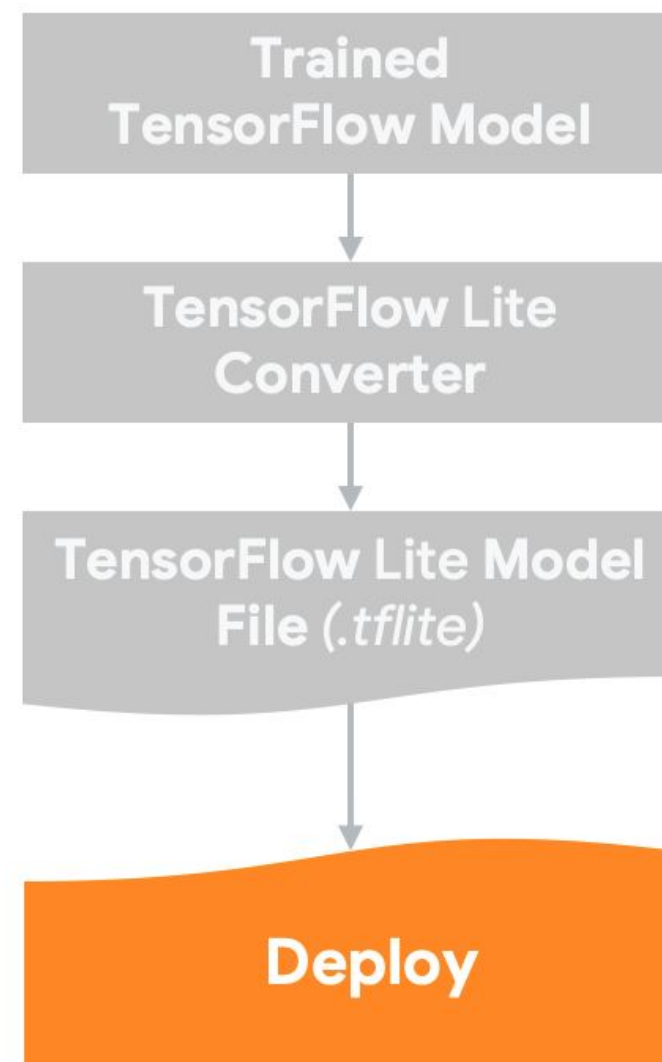
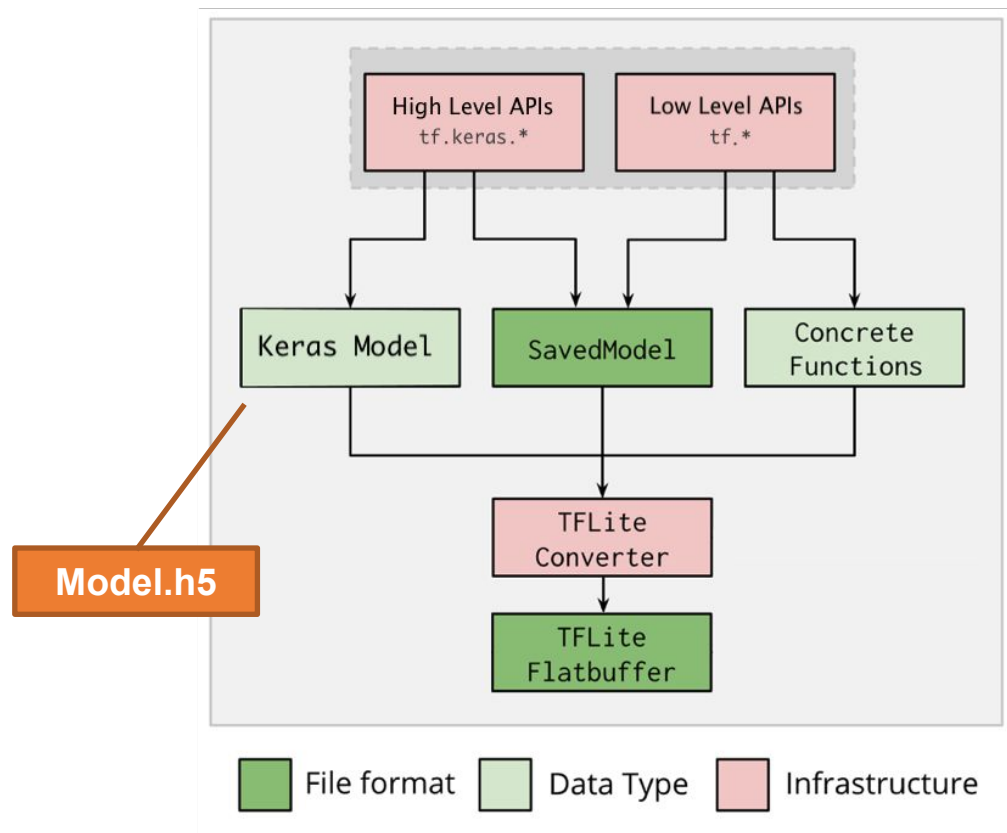
Base Binary Size	3MB+	100KB	~10 KB
Base Memory Footprint	~5MB	300KB	20KB
Optimized Architectures	X86, TPUs, GPUs	Arm Cortex A, x86	Arm Cortex M, DSPs, MCUs



# Optimization and Quantization

Minimizing compression loss

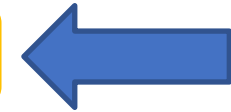
# TensorFlow Workflow



# Converting

Size: 2.1Mb

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



TF Model

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

```
INFO:tensorflow:Assets written to: /tmp/tmpqr8kqp4/assets
```

```
1 # Save .tflite model
```

```
2 open("/content/cifar10.tflite", "wb").write(tflite_model)
```



TFLite Model

Size: .63Mb

```
673324
```

# Converting from a saved model

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

Size: 2.1Mb



**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

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

Size: .63Mb



**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:

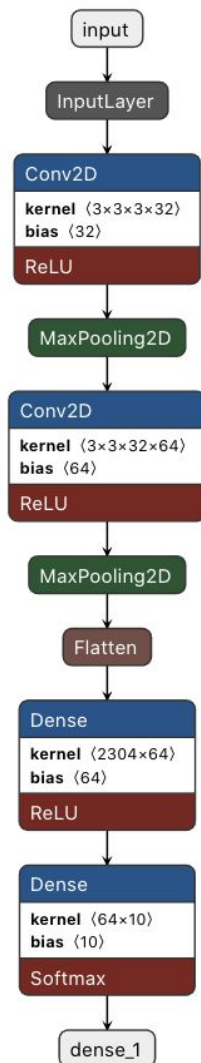
```
[74] 1 converter = tf.lite.TFLiteConverter.from_keras_model(model)
      2 converter.optimizations = [tf.lite.Optimize.DEFAULT]
      3 tflite_quant_model = converter.convert()
      4
```

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

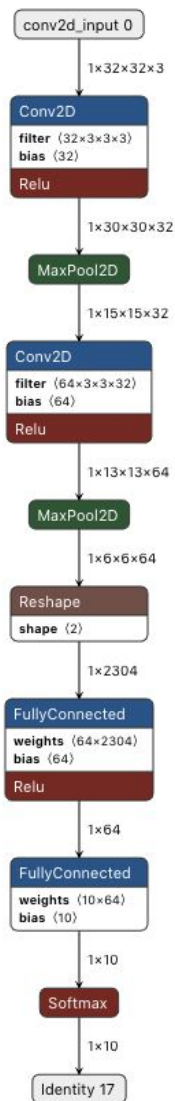
```
[75] 1 # Save .tflite model
      2 open("/content/cifar10_quant.tflite", "wb").write(tflite_quant_model)
```

Size: .18Mb

177232



Cifar\_10.h5



#### NODE PROPERTIES

type FullyConnected ?  
location 5

#### ATTRIBUTES

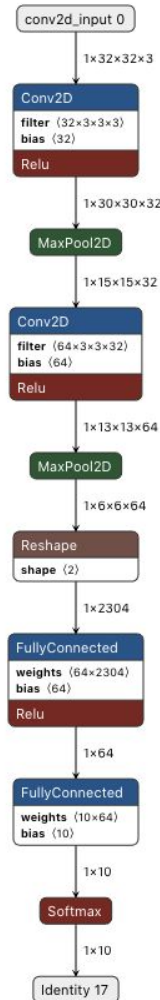
asymmetric\_qua... false +  
fused\_activation... RELU +  
keep\_num\_dims false +  
weights\_format DEFAULT +

#### INPUTS

input name: sequential/flatten/Reshape +  
weights name: sequential/dense/MatMul -  
type: float32[64, 2304]

[  
[  
0.3339117765426636,  
-0.12951909005641937  
-0.19890105724334717  
-0.3362927734851837,  
-0.12596379220485687  
-0.06459484994411469  
0.11760003119707108,  
-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

# Image Classification (inference) using TF-Lite

## Code Time!

CNN\_Cifar\_10\_TFLite.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 \(Coursera\)](#)
- [Text Book: "TinyML" by Pete Warden, Daniel Situnayake](#)

**I want to thank Shawn Hymel and Edge Impulse, Pete Warden and Laurence Moroney from Google, and especially Harvard professor Vijay Janapa Reddi, Ph.D. student Brian Plancher and their staff 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**  
**And stay safe!**



**UNIFEI**