Edge ML のご紹介

~ TensorFlow Lite と Coral を使ってエッジデバイスで機械学習を実装する~

Khanh LeViet

Google
TensorFlow Developer Advocate

@khanhlvg

Agenda

1. Introduction to Edge ML
2. Implement Edge ML (Software)
3. Optimize for edge deployment
4. Implement Edge ML (Hardware)

Introduction to Edge ML

01

Edge ML Explosion

Edge ML Explosion

- Lower latency & close knit interactions

Edge ML Explosion

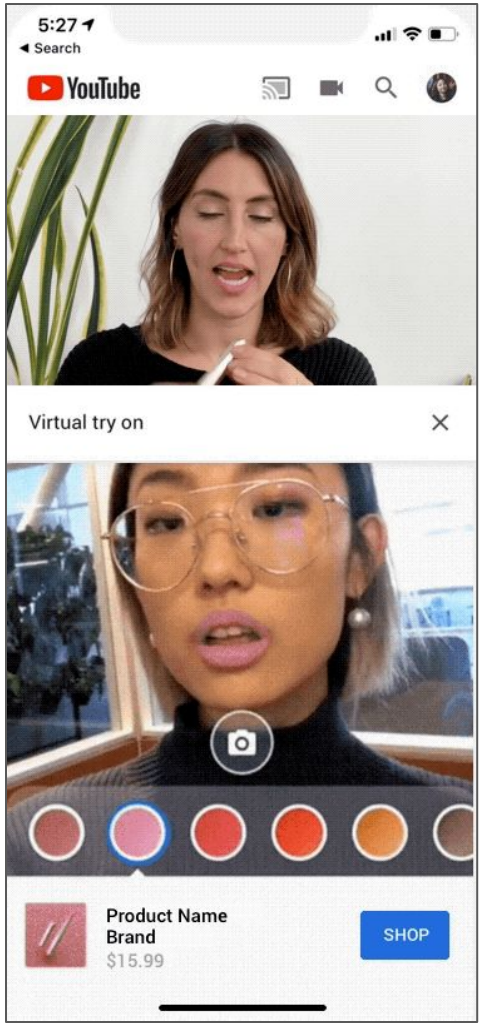
- Lower latency & close knit interactions
- Network connectivity

Edge ML Explosion

- Lower latency & close knit interactions
- Network connectivity
- Privacy preserving

On device ML allows for a new generation of products

YouTube


5:27
Search
YouTube
Virtual try on
Product Name
Brand
$15.99
SHOP

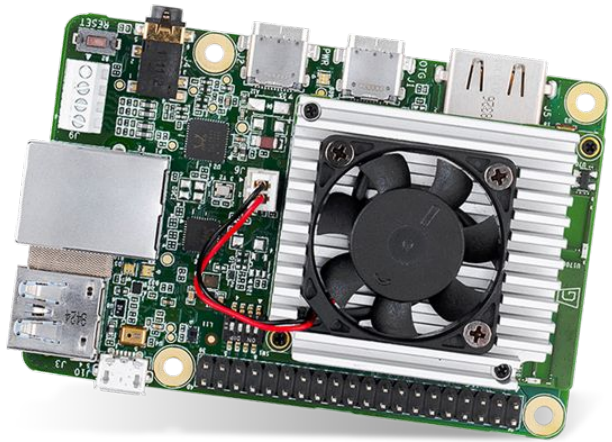
Pixel

Google

Google Assistant

Platforms

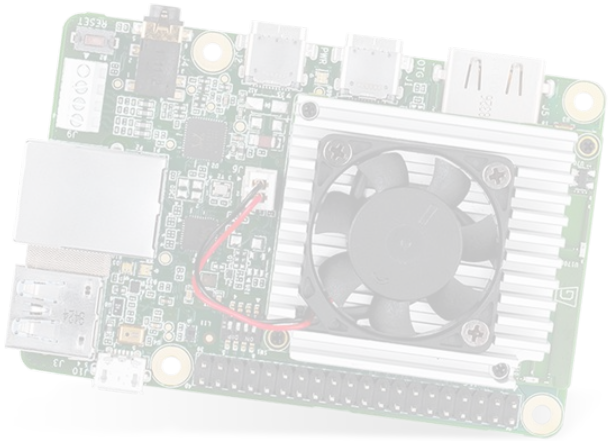
Smartphones



IoT devices

Platforms

Smartphones



IoT devices

1000s of production apps use TFLite globally.

Photos

GBoard

Cloud

YouTube

Assistant

Uber

Hike

Mercari

Viber

Airbnb

Text

Classification
Prediction

Speech

Recognition
Text to Speech
Speech to Text

Image

Object detection
Object location
OCR
Gesture recognition
Facial modelling
Segmentation
Clustering
Compression
Super resolution

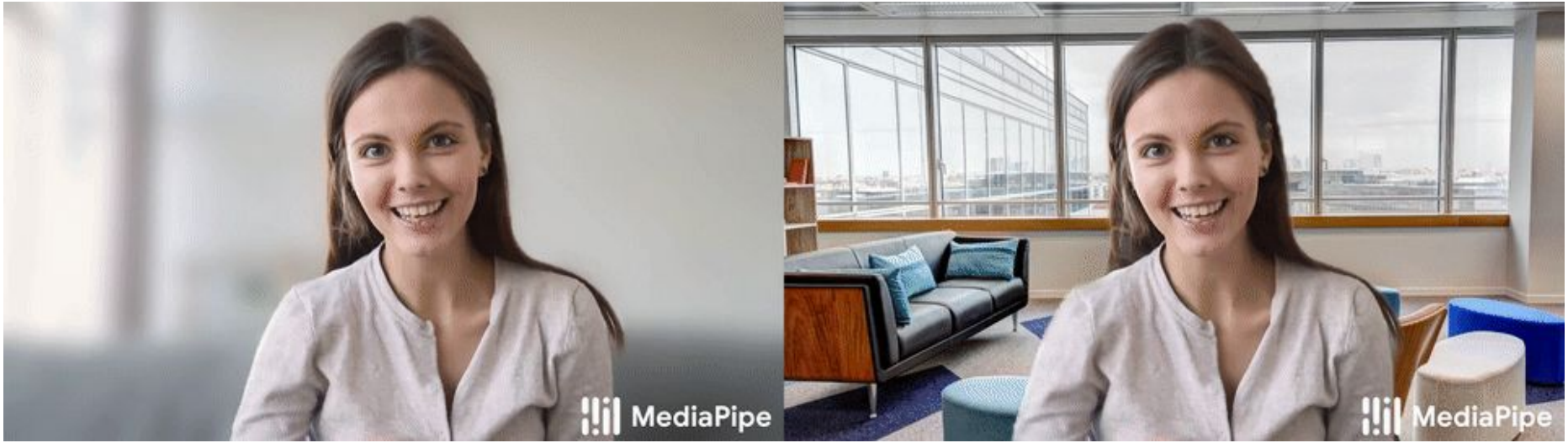
Audio

Translation
Voice synthesis

Content

Video generation
Text generation
Audio generation

Virtual background in video conference

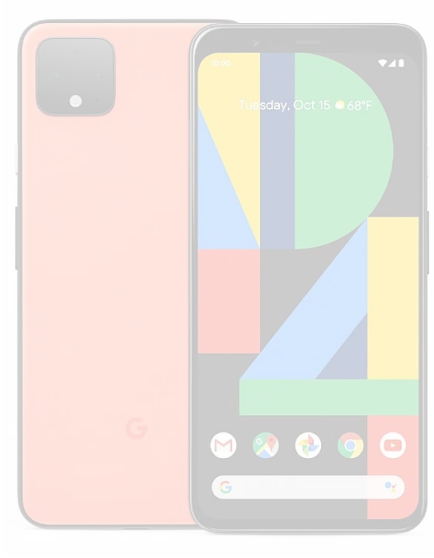


Google Meet

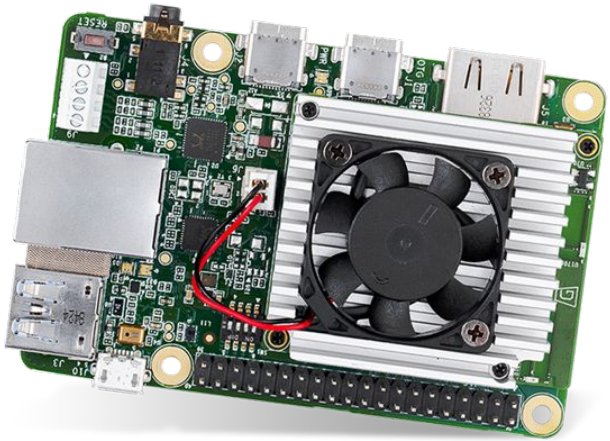
Lookout by Google

An AI toolkit for people with impaired vision, with several helpful features: read aloud text (e.g. money notes, letter), explore surroundings (e.g. walking on the street, shopping) and more.

Platforms



Smartphones



IoT devices

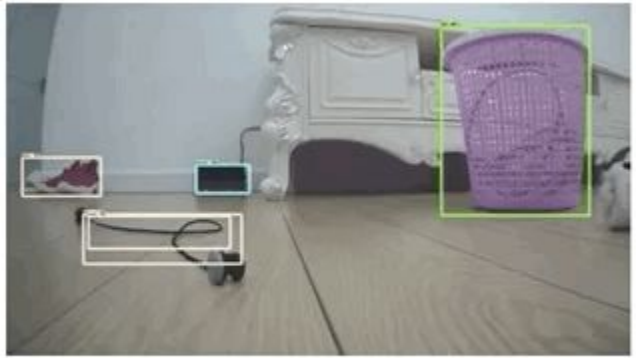
Cleaning robot

by Ecovacs

- Cleaning robot that makes use of AI.

- Use computer vision to detect puffy objects that can't be detected by LIDAR.

- Avoid obstacles when cleaning.

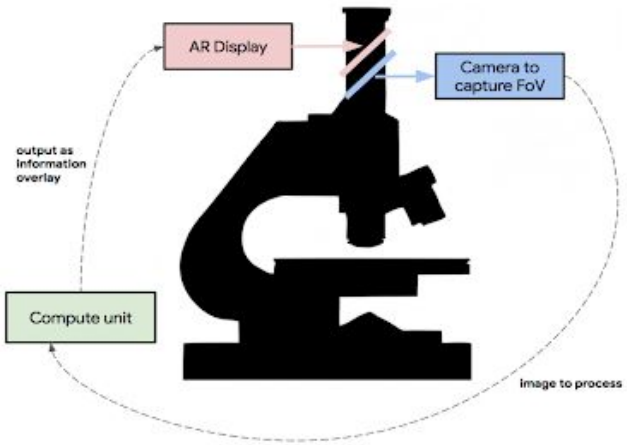


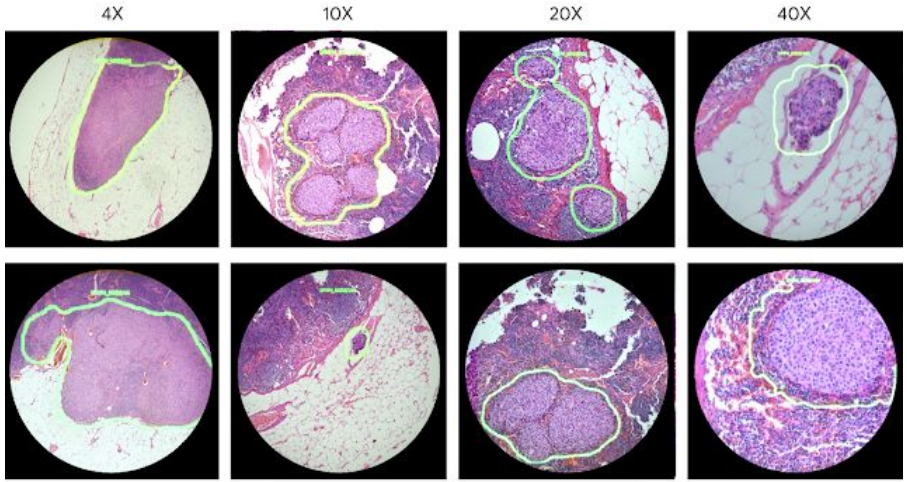
AI Microscope for Cancer Detection

by Google

- Use image segmentation model to detect potential cancer areas in realtime (10fps).

- Accuracy equivalent to a trained pathologist.





Implement Edge ML (Software)

02

Software

TensorFlow Lite

Hardware

Coral

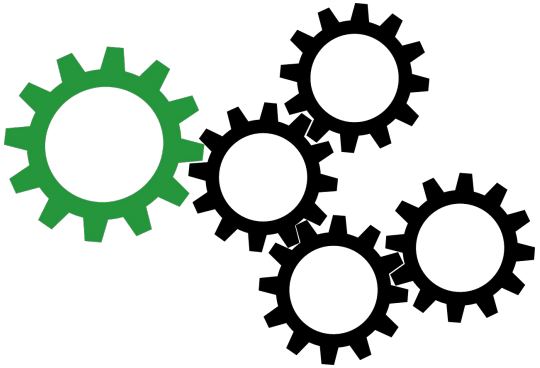
Software

TensorFlow Lite is a framework for deploying ML on **mobile devices and embedded systems**

TensorFlow Lite

Workflow

1. Train a TensorFlow model
2. Convert to TensorFlow Lite format
3. Deploy and run on edge device



Demo
Digit Classifier App

goo.gle/tflite-intro-codelab

Workflow

1. **Train a TensorFlow model**
2. Convert to TensorFlow Lite format
3. Deploy and run on edge device

```python
from tensorflow import keras

# Define the model architecture
model = keras.Sequential([
  keras.layers.InputLayer(input_shape=(28, 28)),
  keras.layers.Dense(32, activation='relu'),
  keras.layers.Dense(10, activation='softmax')
])

# Define how to train the model
model.compile(optimizer='adam', loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the digit classification model
model.fit(train_images, train_labels, epochs=5)
```

Workflow

1. Train a TensorFlow model
2. **Convert to TensorFlow Lite format**
3. Deploy and run on edge device

```python
import tensorflow as tf

# Convert a Keras model to TFLite.
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the TF Lite model.
with tf.io.gfile.GFile('model.tflite', 'wb') as f:
  f.write(tflite_model)
```

```
saved_model_path = "/tmp/saved_models/digit_classifier"

# For TensorFlow 2.0 Keras model use this:
tf.keras.models.save_model(model, saved_model_path)

# For TensorFlow 1.x graph use this:
tf.saved_model.simple_save(session, saved_model_path, inputs, outputs)

# Convert to TensorFlow Lite
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
tflite_model = converter.convert()
open("/tmp/model.tflite", "wb").write(tflite_model)
```

```python
saved_model_path = "/tmp/saved_models/digit_classifier"

# For TensorFlow 2.0 Keras model use this:
tf.keras.models.save_model(model, saved_model_path)

# For TensorFlow 1.x graph use this:
tf.saved_model.simple_save(session, saved_model_path, inputs, outputs)

# Convert to TensorFlow Lite
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
tflite_model = converter.convert()
open("/tmp/model.tflite", "wb").write(tflite_model)
```

Workflow

1. Train a TensorFlow model
2. Convert to TensorFlow Lite format
3. **Deploy and run on edge device (Smartphones)**

```swift
// Create a TFLite interpreter
let interpreter = try Interpreter(modelPath: "model.tflite")

// Allocate memory for the model's input `Tensor`s.
try interpreter.allocateTensors()

// Copy the RGB data to the input `Tensor`.
try interpreter.copy(rgbData, toInputAt: 0)

// Run inference by invoking the `Interpreter`.
try interpreter.invoke()

// Get the output `Tensor` to process the inference results.
let outputTensor = try interpreter.output(at: 0)
```

Workflow

1. Train a TensorFlow model
2. Convert to TensorFlow Lite format
3. **Deploy and run on edge device (IoT devices)**

Install TF Lite via pip

```
pip3 install https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp37-cp37m-linux_armv7l.whl
```

Platform	Python	URL
Linux (ARM 32)	3.5	https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp35-cp35m-linux_armv7l.whl
	3.6	https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp36-cp36m-linux_armv7l.whl
	3.7	https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp37-cp37m-linux_armv7l.whl
Linux (ARM 64)	3.5	https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp35-cp35m-linux_aarch64.whl
	3.6	https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp36-cp36m-linux_aarch64.whl
	3.7	https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp37-cp37m-linux_aarch64.whl

https://www.tensorflow.org/lite/guide/python

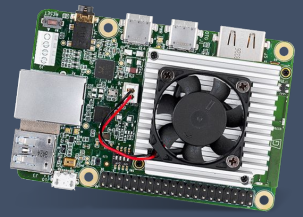
```python
# Create a TFLite interpreter
interpreter = tf.lite.Interpreter(model_path='model.tflite')

# Allocate memory for the model's input `Tensor`s.
interpreter.allocate_tensors()

# Copy the RGB data to the input `Tensor`.
interpreter.set_tensor(0, rgbData)

# Run inference by invoking the `Interpreter`.
interpreter.invoke()

# Get the output `Tensor` to process the inference results.
output = style_bottleneck = interpreter.tensor(100)()
```



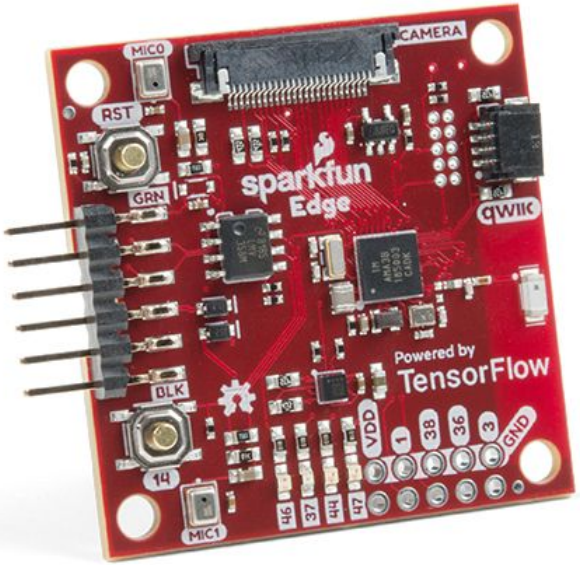
Python (IoT)

Optimize for edge deployment

03

Optimization is important

- Limited compute power
- Limited memory
- Battery consumption
- App size



Methods

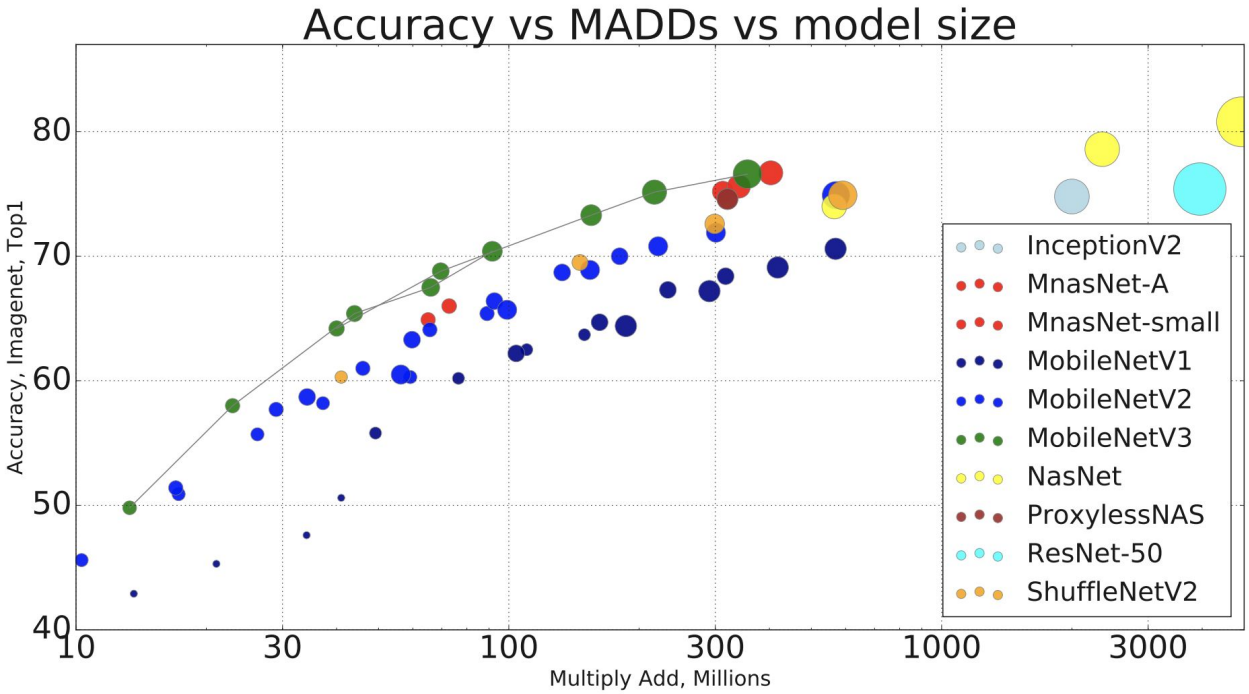
1. Use mobile optimized model architecture
2. Apply quantization
3. Apply pruning
4. Use hardware accelerator

Model Architecture

	Inception v4	Mobilenet_V3_1.0_224	
Top-1 Accuracy	80.0%	75.7%	-6%
Top-5 Accuracy	95.0%	92.8%	-3%
TF Lite Inference Latency*	320ms	8.58ms	**37x faster**
Model Size	170.7MB	22.0MB	**7.8x smaller**

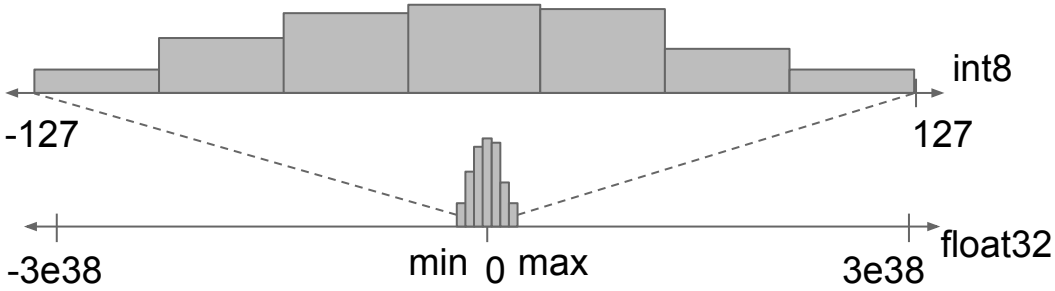
** **Pixel 4** - 4 Threaded CPU, Oct 2020*

Model Architecture



Source: Andrew Howard & Mark Sandler et.al., Searching for MobileNetV3 (2019)

Quantization



Reduce number of bits for model weights and activations

Quantization

Model	Float baseline	Post-training quantization	Accuracy drop
Mobilenet v1	71.03%	69.57%	1.46 p.p
Mobilenet v2	70.77%	70.2%	0.57 p.p
Resnet v1	76.3%	75.95%	0.05 p.p

Ease of use vs. accuracy vs. latency

Tradeoffs between technique

Approach	Technique	Ease of use	Accuracy loss	CPU speed up	Hardware
Post Training	Dynamic Range Quantization	No data needed	small	2~3x	CPU
	Integer Quantization	Unlabeled data	small	2~4x	CPU, DSP/TPU
During Training	Quantization Aware Training	Training	smaller	2~4x	CPU, DSP/TPU

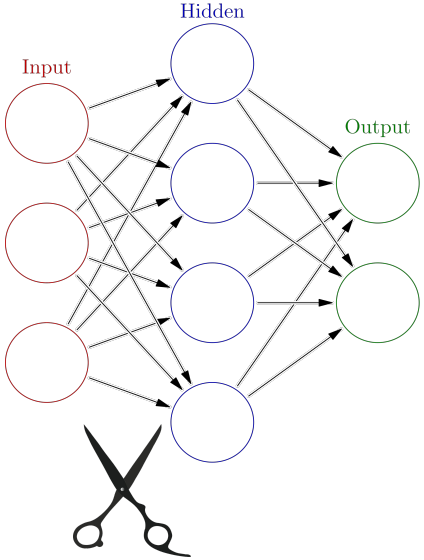
Apply dynamic range quantization

```
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_quant_model = converter.convert()
```

Pruning

What does it mean?

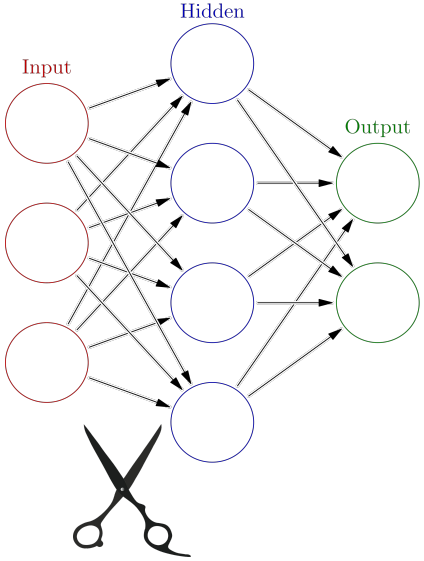
- Drop connections during training.
- Dense tensors will now be sparse (filled with zeros).



Pruning

Benefits

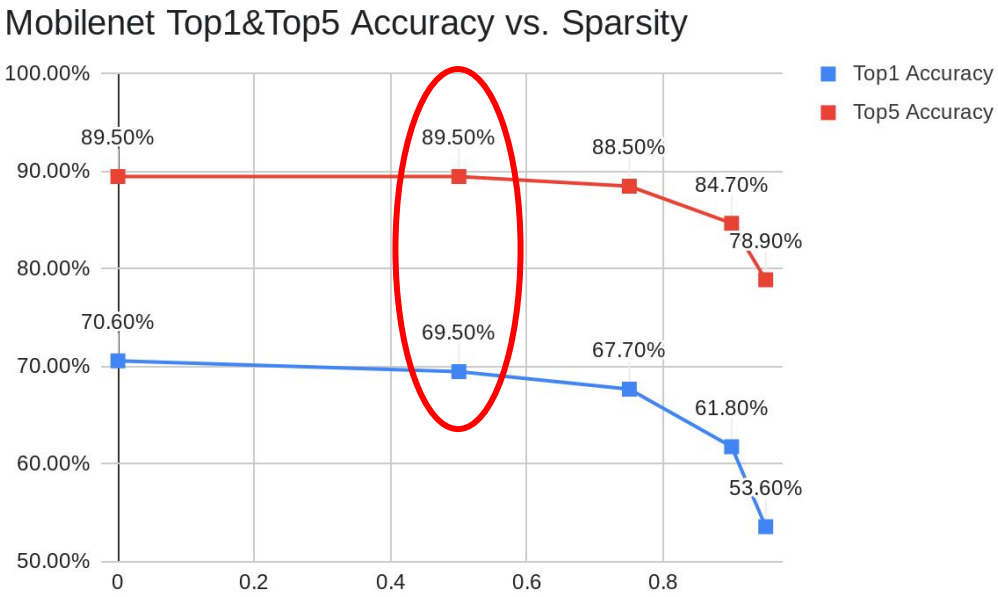
- **Smaller models.** Sparse tensors can be compressed.
- **Faster models.** Less operations to execute.



Pruning

Result

- **Negligible** accuracy **loss** at **50%** sparsity
- **Small** accuracy **loss** at **75%**



Apply pruning

```
model = model = tf.keras.Sequential([...])

# Define pruning parameters.
pruning_params = {
  'pruning_schedule': tfmot.sparsity.keras.PolynomialDecay(
      initial_sparsity=0.50, final_sparsity=0.80,
      begin_step=0, end_step=end_step)
}

# Prune model
model = prune_low_magnitude(model, **pruning_params)

# Recompile and train
model.compile(...)
model.fit(...)
```

Hardware accelerator

CPU 37 ms	CPU 2.8x 13 ms	GPU 5.2x 7 ms	EdgeTPU 11.6x 3.2 ms	Hexagon DSP 13.2x 2.8 ms
Floating point	Quantized Fixed-point	OpenCL Float16	Quantized Fixed-point	Quantized Fixed-point

Mobilenet V1-1.0-224

Pixel 4 *- Single Threaded CPU, June 2020*

Supported accelerators

- Android:
  - GPU
  - Hexagon DSP
  - NN-API

- iOS:
  - GPU
  - CoreML

Inference w/ GPU

JAVA

```java
GpuDelegate delegate = new GpuDelegate();
Interpreter.Options options = new Interpreter.Options().addDelegate(delegate);

try (Interpreter tflite = new Interpreter(awesomeModel, options)) {
  tflite.run(inputs, outputs);
}

// cleanup
delegate.close();
```

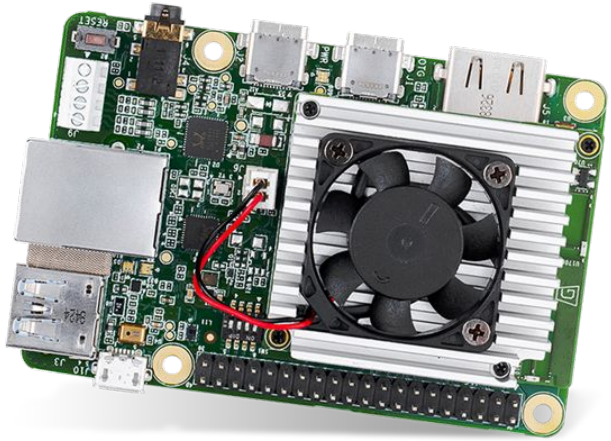
Implement Edge ML (Hardware)

04

Hardware

Coral

Hardware accelerator



Coral products

Prototyping

Dev Board

A single-board computer with a removable system-on-module (SOM) featuring the Edge TPU.

Price
$129.99

USB Accelerator

A USB accessory featuring the Edge TPU that brings ML inferencing to existing systems.

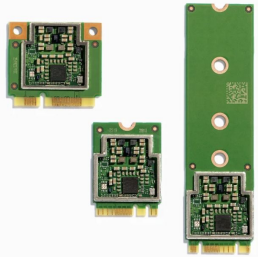
Price
$59.99

Production

System-on-Module

A fully integrated System on Module in a 40mm x 48mm pluggable module.

Price
$114.99



PCI-E Accelerators

PCI-E device for easy integration of Edge TPU into existing systems. A/E, B/M, M.2

Price
$34.99

Hardware accelerator

the same chip as

Coral

CPU 37 ms	CPU 2.8x 13 ms	GPU 5.2x 7 ms	EdgeTPU 11.6x 3.2 ms	Hexagon DSP 13.2x 2.8 ms
Floating point	Quantized Fixed-point	OpenCL Float16	Quantized Fixed-point	Quantized Fixed-point

Mobilenet V1-1.0-224

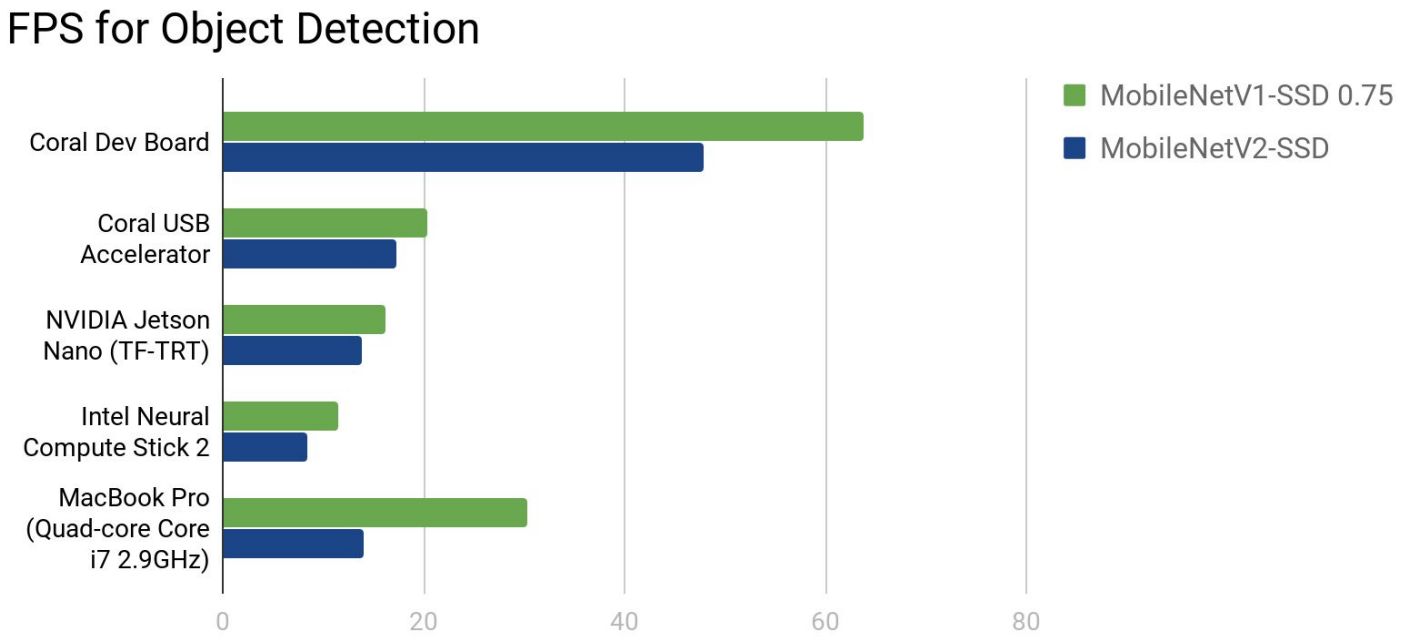
Pixel 4 *- Single Threaded CPU, June 2020*

Other options

NVIDIA Jetson Nano
$99

Intel Neural Compute Stick
$99

Comparison



Device for USB accelerators: Raspberry Pi 3 B+
Source: https://medium.com/@aallan/benchmarking-edge-computing-ce3f13942245

Thank you

Learn more about TensorFlow Lite at
https://www.tensorflow.org/lite

Khanh LeViet

Developer Advocate

@khanhlvg