模型优化工具包

一套用于优化模型的 TensorFlow 和 TensorFlow Lite 的工具包，使模型更精简、快速，以便于推断。

优先考虑在各种模型和硬件上的普遍适用性。

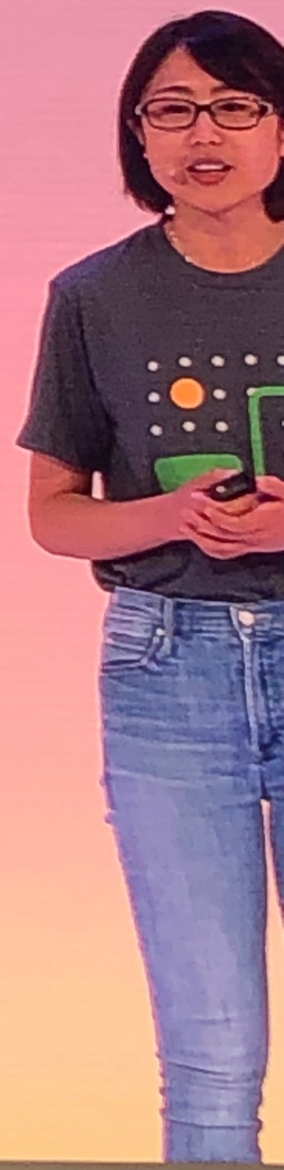
https://tensorflow.google.cn/model_optimization

模型优化工具包

当前 API

训练后：基于 TensorFlow Lite 转换器

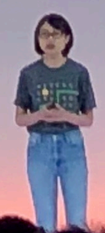
训练期间：基于 tf.keras



模型优化工具包

训练期间 API

1. 使用 tf.keras 构建模型
2. 将训练期间 API 应用到 tf.keras 层级或整个模型
3. 在 TensorFlow 或 TensorFlow Lite 中导出模型以用于推断

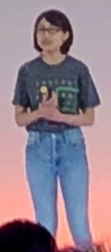


量化

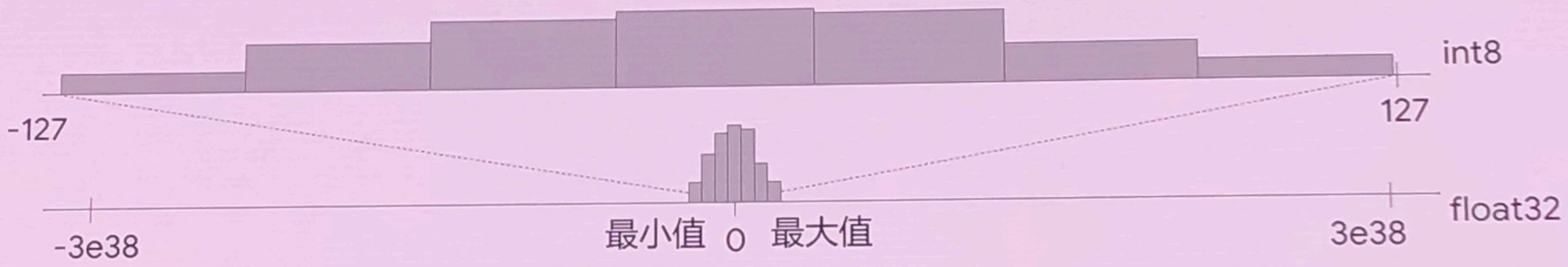
是什么?

降低模型参数的精度（例如网络权重）。

以更低的精度在静态参数和动态输入/激活值之间进行运算。
即 float32 -> int8



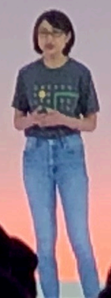
均匀/线性量化



范围 =（最大值 - 最小值）/ 2 ^ 位数

量化值 = 浮点值 / 范围

浮点值 = 量化值 * 范围



量化

为何很难?

有损转换影响了准确率。

需要硬件支持。

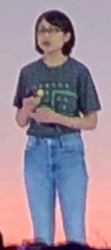
量化有很多不同的类型。

整数量化需要浮点张量范围统计数据。

易用性、准确率、延迟性对比

工具权衡

技术	易用性	准确率	延迟性
“混合”量化	无需数据	准确率降低	比浮点值快
整数量化	无标签数据	准确率降低较少	最快
量化感知训练	有标签训练数据	准确率降低最少	最快



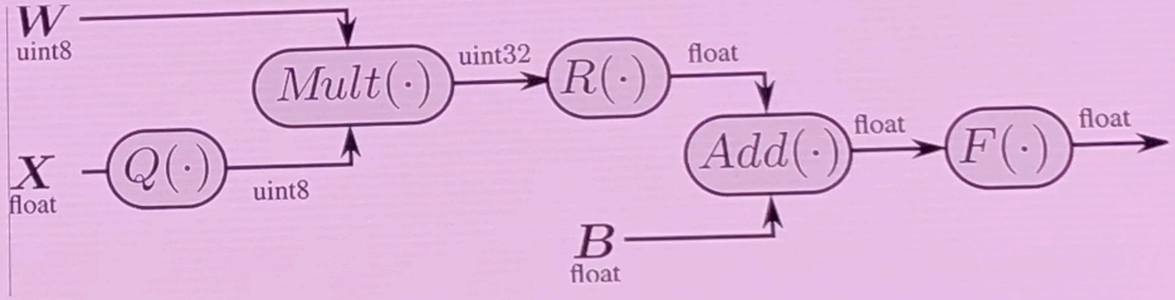
“混合”量化

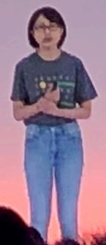
是什么?

整型权重

浮点激活值

在推断期间计算最小值/最大值





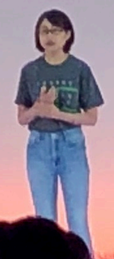
“混合”量化

结果

模型参数减少 4 倍（32 位 -> 8 位）

卷积模型执行速度提升 10-50%（CPU/浮点数支持）

全连接模型和 RNN 模型速度提升高达 3 倍（CPU/浮点数支持）



通过训练后（混合）量化进行模型转换

```
import tensorflow as tf

saved_model_dir = "/path/to/mobilenet_v1_1.0_224/"
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()

open("converted_model.tflite", "wb").write(tflite_model)
```

整数量化

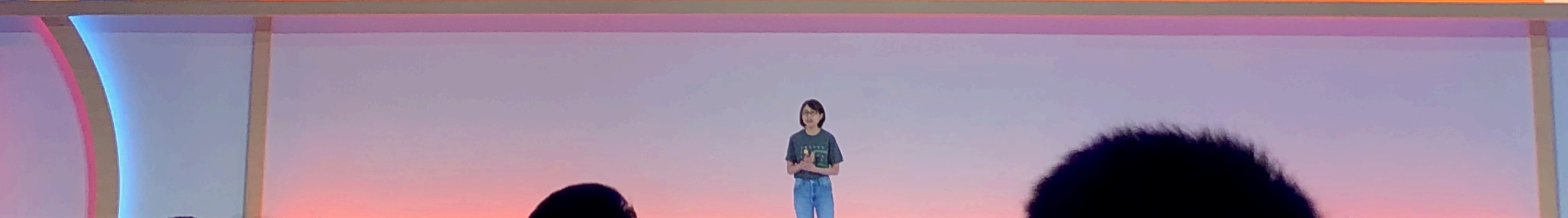
是什么?

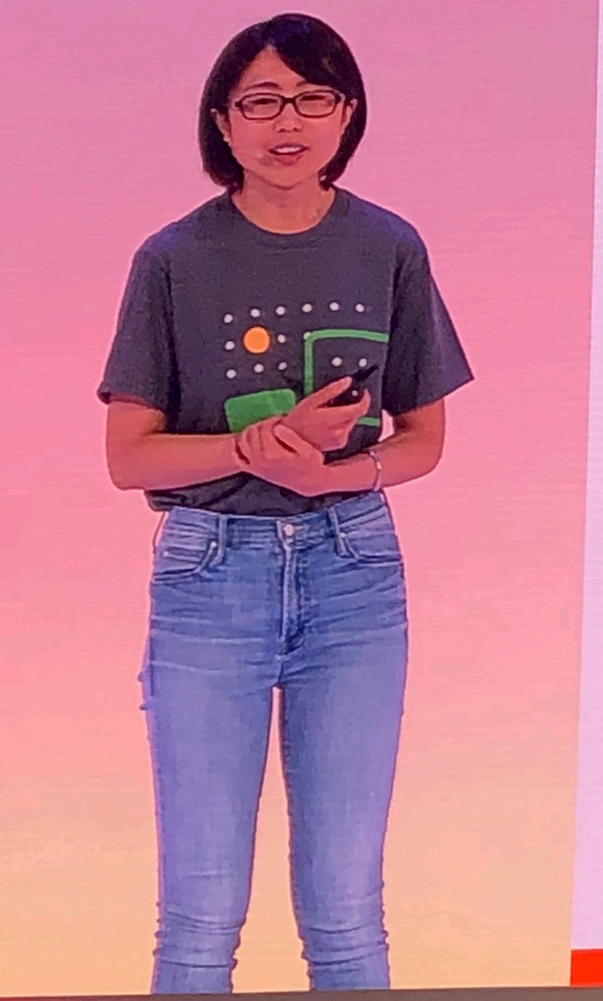
整型权重

整型激活值

所有运算都是整型运算

在模型转换期间计算最小值/最大值





整数量化

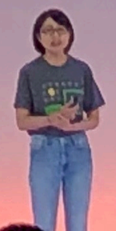
结果

模型参数减少 4 倍（32 位 -> 8 位）

卷积模型执行速度提升 10-50%

全连接模型和 RNN 模型在 CPU 上的速度提升 2-4 倍

支持硬件加速器 (NNAPI/EdgeTPU)

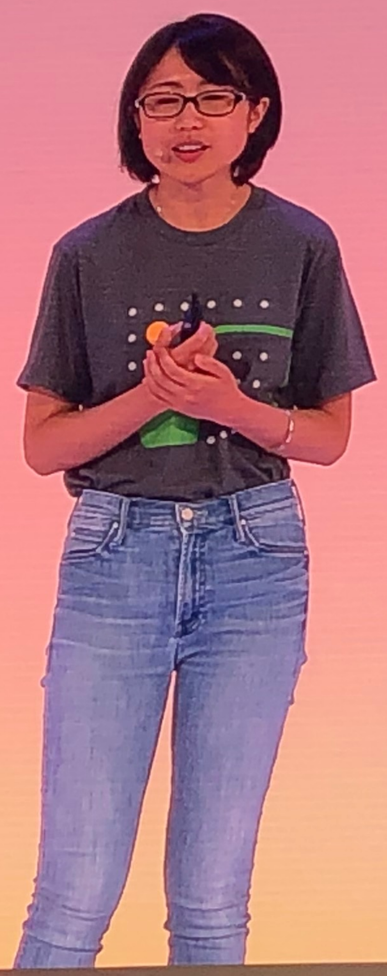


通过训练后（整数）量化进行模型转换

```
import tensorflow as tf

saved_model_dir = "/path/to/mobilenet_v1_1.0_224/"
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
def data_generator():
  for i in range(calibration_steps):
    # get sample input data
    yield [input_sample]
converter.representative_dataset = data_generator
tflite_model = converter.convert()
open("converted_model.tflite", "wb").write(tflite_model)
```

RepresentativeDataset 示例

```
import tensorflow_datasets as tfds

def representative_dataset_gen():
  def preprocess_map_fn(example):
    image = preprocess_fn(example['image'])
    return tf.reshape(image, (1, FLAGS.image_size, FLAGS.image_size, 3))

  data = tfds.load('imagenet2012')
  data = data.map(preprocess_map_fn)
  for _ in range(FLAGS.num_calibration_steps):
    image, = data.take(1)
    yield [image]
```



量化感知训练

正在开发中

提高敏感模型的准确率。

在训练期间模拟量化误差。

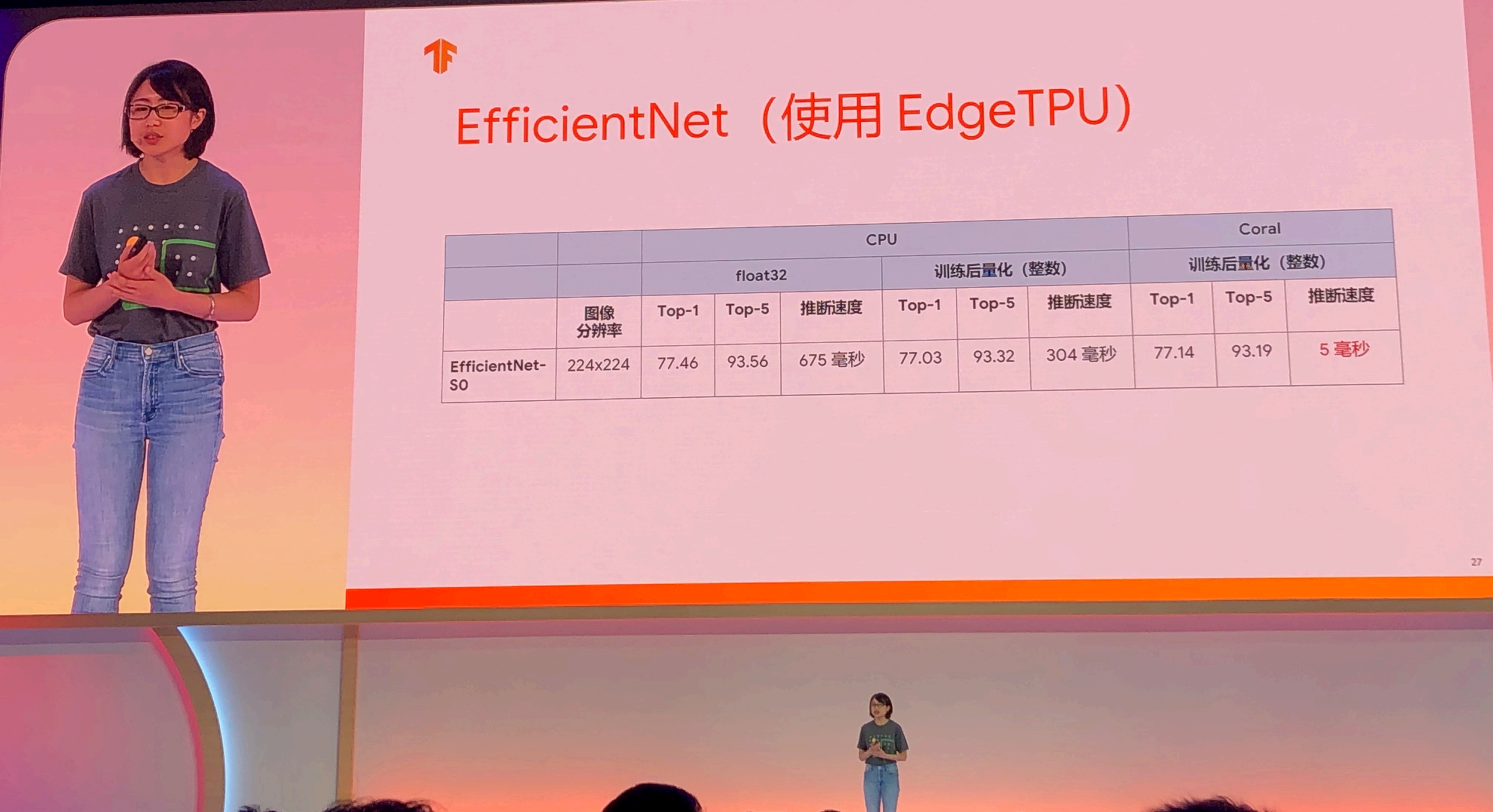
作为 tf.keras API 提供。

训练后整数量化
准确率
模型
浮点数基准
训练期间量化
训练后量化
Mobilenet v1 1.0 224
70.95%
69.97%
69.54%
Resnet v2
76.80%
76.60%
Inception v3
77.90%
77.50%
77.70%
25

训练后整数量化

准确率

模型	浮点数基准	训练期间量化	训练后量化
语音识别	*	-	0.1% 增量（使用浮点数）
MobileSSD_V2	22.1%	21.7%	17.4%
MNasNet_1.0_224	74.46 / 92.12%	-	73.98 / 91.96%
EfficientNet (224x224)	77.46 / 93.56 %	-	77.14 / 93.19 %


EfficientNet（使用 EdgeTPU）
CPU
Coral
float32
训练后量化（整数）
训练后量化（整数）
图像分辨率
Top-1
Top-5
推断速度
Top-1
Top-5
推断速度
Top-1
Top-5
推断速度
EfficientNet-S0
224x224
77.46
93.56
675 毫秒
77.03
93.32
304 毫秒
77.14
93.19
5 毫秒
27

8 位整型量化规范

面向硬件供应商和内核实现人员的操作者规范。

https://tensorflow.google.cn/lite/performance/quantization_spec

Float16 推断

最新发布

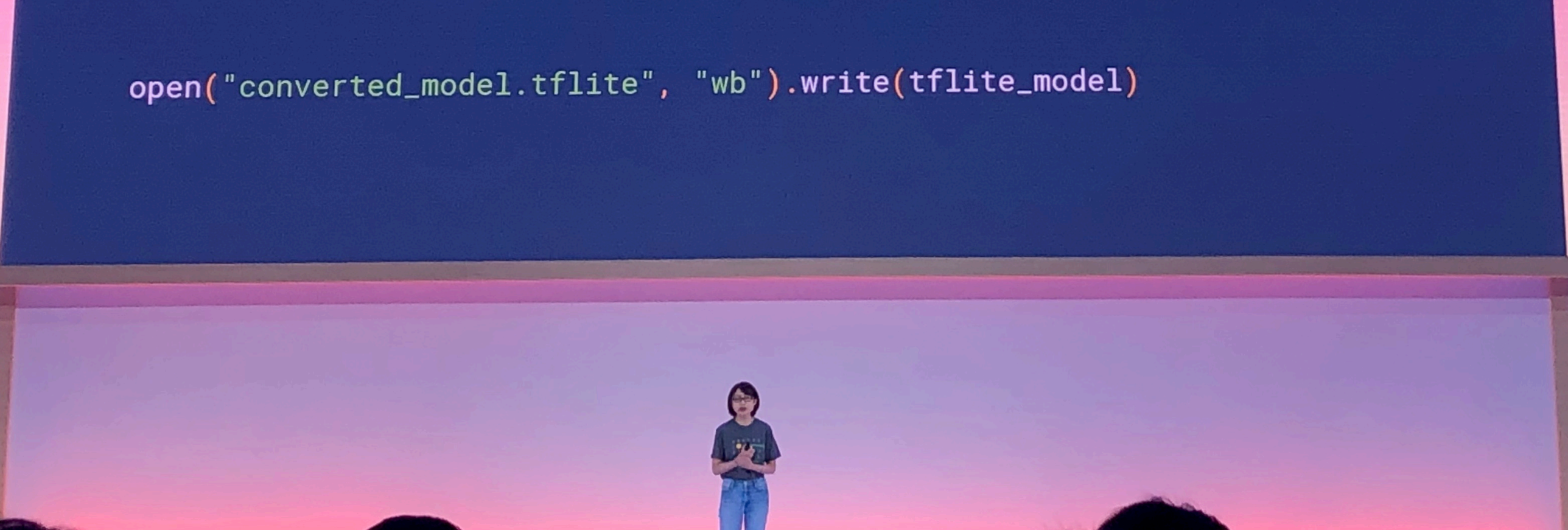
无需最小值/最大值数据

float16 权重压缩

可以利用 GPU 进行加速

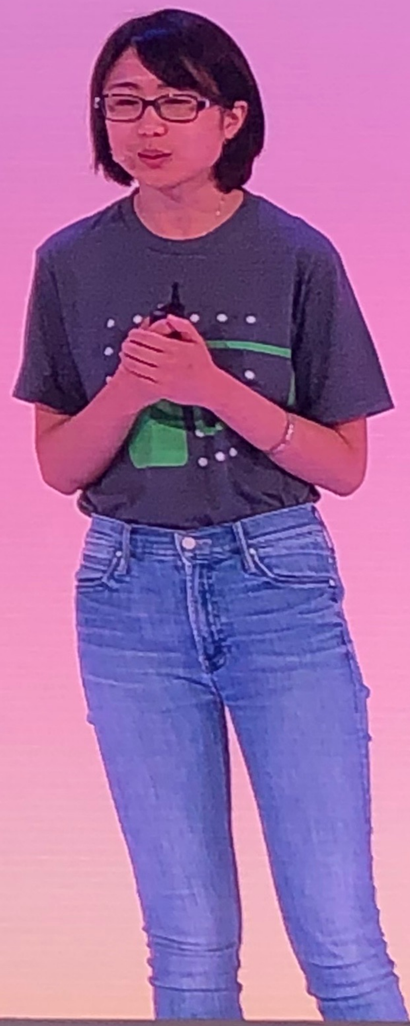
通过 float16 权重进行模型转换（正在开发中）

```
import tensorflow as tf

saved_model_dir = "/path/to/mobilenet_v1_1.0_224/"
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec = tf.lite.TargetSpec(supported_types=[tf.float16])
tflite_model = converter.convert()

open("converted_model.tflite", "wb").write(tflite_model)
```



训练后工具摘要

技术	优势	硬件
训练后“混合”量化	精简 4 倍 速度提升 2-3 倍	CPU
训练后整数量化	精简 4 倍 速度提升幅度更大	CPU/DSP/EdgeTPU/NNAPI 等
训练后 fp16 量化	精简 2 倍 潜在 GPU 加速	CPU/GPU



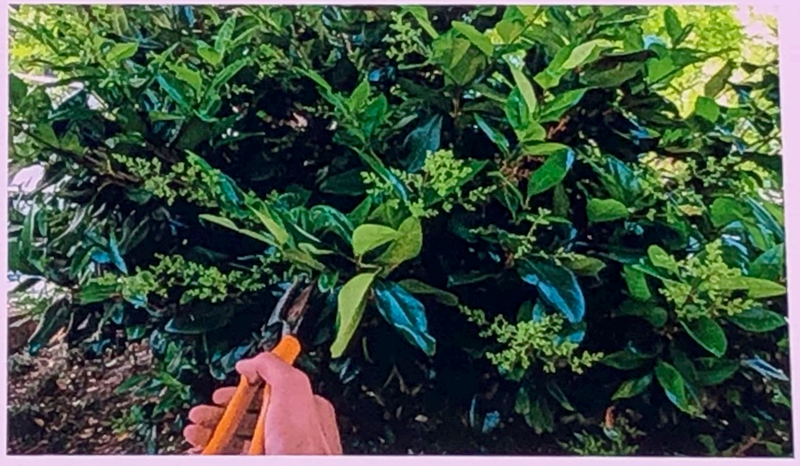
剪枝

是什么意思？

在训练期间删除连接。

密集张量现在将变得稀疏（用零填充）。

可以通过结构化块 (nxm) 或随机 (1x1) 删减连接。



剪枝并非Dropout！

剪枝会执行什么操作？

在训练期间根据大小（而非随机）删除连接。

改变权重张量，但不改变动态激活值。

剪枝

稀疏性的优势

二进制文件变小。

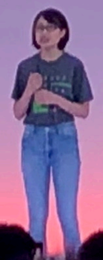
模型更小，可以减少内存带宽消耗量。

可以针对某些块配置的 CPU 和自定义硬件实现更快的内核。

剪枝

当前端到端流程

1. 构建 tf.keras 训练模型。
2. 使用训练期间剪枝 API 训练模型。
3. 生成的权重将包含许多零值。
4. 使用文件压缩库（gzip、bzip 等）进行压缩。



剪枝/稀疏性训练 API

```
import tensorflow_model_optimization as tfmot

model = build_your_model()

pruning_schedule = tfmot.sparsity.keras.PolynomialDecay(
                        initial_sparsity=0.0, final_sparsity=0.5,
                        begin_step=2000, end_step=4000)

model_for_pruning = tfmot.sparsity.keras.prune_low_magnitude(model,
pruning_schedule=pruning_schedule)

...

model_for_pruning.fit(...)
```

剪枝

模型适用范围

音频/语音处理：关键字检索、说话人验证、文字转语音、waveRNN 等。

图像处理：inceptionv3、mobilenetv1、mobilenetv2 等。

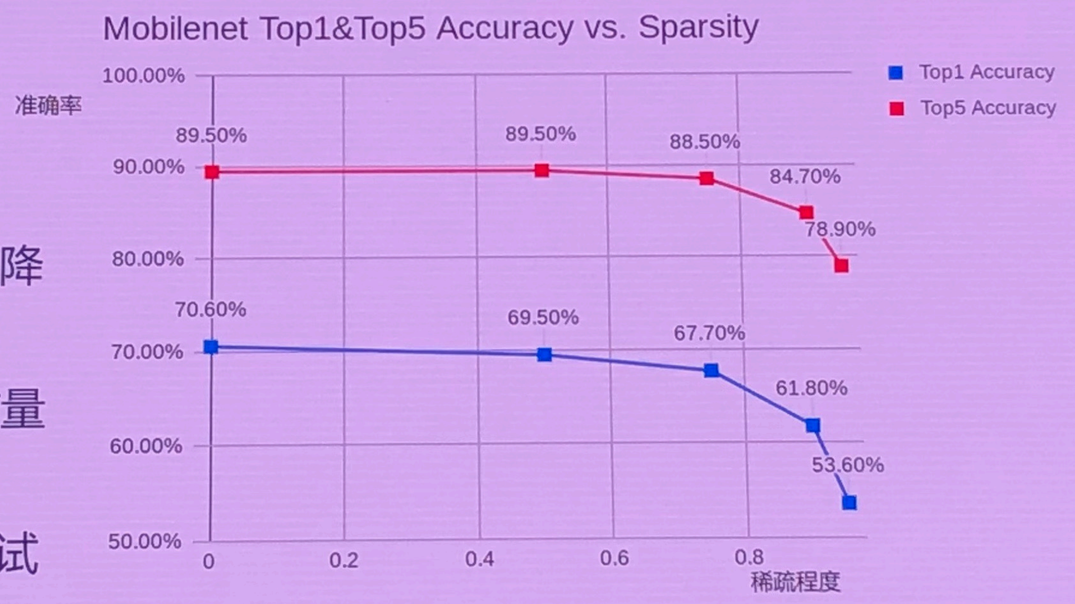
剪枝

结果

50-80% 稀疏性，准确率降低幅度可忽略不计。

独立于量化技巧，通常与量化技巧的配合效果不错。

您可以通过 keras 微调尝试不同的参数组合。



要点

机器学习开发者

现已提供以下功能：

1. 开启 TensorFlow Lite 转换器的默认优化功能。
2. 提供有代表性的数据集以进行整数量化。
3. 如果您希望在GPU上加速模型，使用float16优化。
4. 如果您要缩减静态模型的大小，可以使用剪枝 API 训练模型。

要点

机器学习开发者

以下功能正处于积极开发中：

1. 量化感知训练 tf.keras API
2. 快速稀疏内核和稀疏张量表示法
3. 在现有整数量化技术的基础上进一步优化

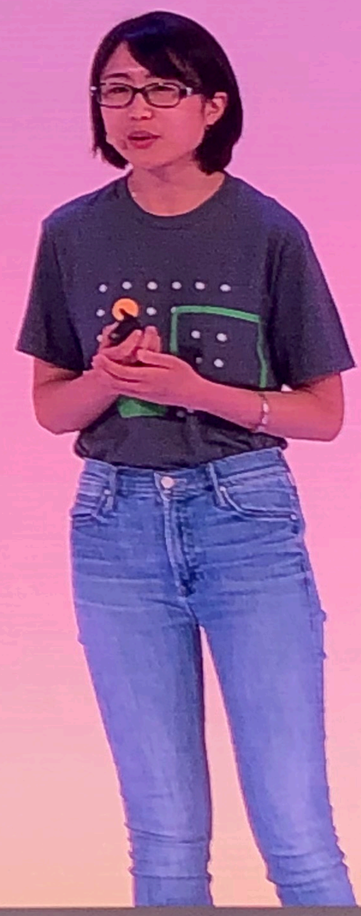
要点

硬件供应商

面向硬件供应商的操作者规范。

https://tensorflow.google.cn/lite/performance/quantization_spec

如有需求，请联系 tflite@tensorflow.org



谢谢！

https://tensorflow.google.cn/model_optimization

https://tensorflow.google.cn/lite/performance/model_optimization

向我们提供反馈：

https://github.com/tensorflow/tensorflow/issues

https://github.com/tensorflow/model-optimization/issues