TensorFlow Lite

Lightweight cross-platform solution for mobile and embedded devices
Why TensorFlow Lite?
ML runs in many places

- Access to more data
- Fast and closely knit interactions
- Privacy preserving
Creates many challenges

- Reduced compute power
Creates many challenges

- Reduced compute power
- Limited memory
Creates many challenges

- Reduced compute power
- Limited memory
- Battery constraints
Simplifying ML on-device

TensorFlow Lite makes these challenges much easier!
What can I do with it?
Many use cases

- **Text**
  - Classification
  - Prediction
  - Recognition
  - Text to Speech
  - Speech to Text

- **Speech**
  - Translation
  - Voice Synthesis

- **Image**
  - Object detection
  - Object Location
  - OCR
  - Gesture recognition
  - Facial modelling
  - Segmentation
  - Clustering
  - Compression
  - Super Resolution

- **Audio**
  - Video generation
  - Text generation
  - Audio generation

- **Content**
Who is using it?
>2B mobile devices

Have TensorFlow Lite deployed on them in production
Some of the users ...
Google Assistant is on 1B+ devices

Wide range of devices: High/low end, arm, x86, battery powered, plugged in, many operating systems

Phones  Speakers  Smart Displays  Cars

TVs  Laptops  Wearables  Others
Key Speech On-Device Capabilities

- “Hey Google” Hotword with VoiceMatch
  - Tiny memory and computation footprint, running continuously
  - Extremely latency sensitive

- On-device speech recognition
  - High computation running in shorter bursts
Online Education Brand with the largest numbers of users in China

800 million
Users in total

22 million
DAU
Youdao Applications with TensorFlow Lite

Youdao Dictionary  Youdao Translator  U-Dictionary
Youdao On-Device AI Translation & OCR

- Applied in Youdao dictionary and translator apps
- Offline photo translation speed improved 30-40%
- Support Realtime AR translation
Model conversion

The conversion flow to TensorFlow Lite is simple ...

TensorFlow (estimator or Keras) → Saved Model → TF Lite Converter → TF Lite Model
Model conversion

... however there are points of failure

- Limited ops
- Unsupported semantics (e.g. control-flow in RNNs)
Model conversion

TensorFlow Select

Available now
- Enables **hundreds more ops** from TensorFlow on CPU.
- Caveat: binary size increase (~6MB compressed).

In the pipeline
- Selective registration
- Improved performance
Model conversion

Control flow support

In the pipeline

Control flow are core to many ops (e.g. RNNs) and graphs. Thus we are adding support for:

- Loops
- Conditions
Inference performance

**CPU**
- 124 ms

**CPU 1.9x**
- 64 ms

**GPU 7.7x**
- 16 ms

**Edge TPU 62x**
- 2 ms

- CPU on MobileNet V1
- CPU w/ Quantization
- Flow OpenGL 16
- Quantized Fixed-point

**MobileNet V1**
Benchmarking

Benchmarking and profiling

Available

Improvements to the Model Benchmark tool:

- Support for threading
- Per op profiling
- Support for Android NN API
Benchmarking
Per-op profiling breakdown

<table>
<thead>
<tr>
<th>node type</th>
<th>start</th>
<th>first</th>
<th>avg ms</th>
<th>[ ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONV_2D</td>
<td>0.000</td>
<td>4.269</td>
<td>4.269</td>
<td>0.107%</td>
</tr>
<tr>
<td>DEPTHWISE_CONV_2D</td>
<td>4.270</td>
<td>2.150</td>
<td>2.150</td>
<td>0.054%</td>
</tr>
<tr>
<td>CONV_2D</td>
<td>6.421</td>
<td>6.107</td>
<td>6.107</td>
<td>0.153%</td>
</tr>
<tr>
<td>DEPTHWISE_CONV_2D</td>
<td>12.528</td>
<td>1.366</td>
<td>1.366</td>
<td>0.034%</td>
</tr>
<tr>
<td>RESHAPE</td>
<td>79.440</td>
<td>0.002</td>
<td>0.002</td>
<td>0.000%</td>
</tr>
<tr>
<td>SOFTMAX</td>
<td>79.443</td>
<td>0.029</td>
<td>0.029</td>
<td>0.001%</td>
</tr>
</tbody>
</table>
### Benchmarking

### Profiling summary

**Number of nodes executed:** 31

<table>
<thead>
<tr>
<th>Node type</th>
<th>Count</th>
<th>Avg ms</th>
<th>Avg %</th>
<th>CDF %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONV_2D</td>
<td>15</td>
<td>1.406</td>
<td>89.27%</td>
<td>89.27%</td>
</tr>
<tr>
<td>DEPTHWISE_CONV_2D</td>
<td>13</td>
<td>0.169</td>
<td>10.73%</td>
<td>100.00%</td>
</tr>
<tr>
<td>SOFTMAX</td>
<td>1</td>
<td>0.000</td>
<td>0.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>RESHAPE</td>
<td>1</td>
<td>0.000</td>
<td>0.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>AVERAGE_POOL_2D</td>
<td>1</td>
<td>0.000</td>
<td>0.00%</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

Timings (microseconds): count=50 first=79449 curr=81350 min=77385 max=88213 avg=79732 std=1929

Memory (bytes): count=0

31 nodes observed

Average inference timings in us: Warmup: 83235, Init: 38467, no stats: 79760.9
What is a delegate?
Fast execution

Android Neural Network API delegate

*Enables hardware supported by the Android NN API*
Fast execution

GPU delegate

Preview available!

- 2–7x faster than the floating point CPU implementation
- Adds ~250KB to binary size (Android/iOS).
Fast execution

GPU delegate

In the pipeline

● Expand coverage of operations
● Further optimize performance
● Evolve and finalize the APIs

Make it generally available!
Fast execution

Edge-TPU delegate

Enables next generation ML hardware!

- High performance
- Small physical and power footprint

Available in Edge TPU development kit
Optimization

Make your models even smaller and faster.
## Optimization

<table>
<thead>
<tr>
<th>Available</th>
<th>In the pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Quantization</strong></td>
<td>Keras-based quantized training (CPU/NPU)</td>
</tr>
<tr>
<td>Post-training quantization (CPU)</td>
<td>Post-training quantization (CPU/NPU)</td>
</tr>
<tr>
<td><strong>Other optimizations</strong></td>
<td>Keras-based connection pruning</td>
</tr>
<tr>
<td>Model optimization toolkit</td>
<td></td>
</tr>
</tbody>
</table>
Optimization

Quantization

New tools

- Post-training quantization with float & fixed point
- Great for CPU deployments!
Optimization

Quantization

Benefits

● 4x reduction in model sizes
● Models, which consist primarily of convolutional layers, get 10–50% faster execution (CPU)
● Fully-connected & RNN-based models get up to 3x speed-up (CPU)
Optimization
Quantization

In the pipeline

- Training with quantization Keras-based API
- Post-training quantization with fixed point math only

*Even better performance on CPU*

*Plus enable many NPUs!*
Keras-based quantization API

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([tf.keras.layers.Flatten(),
                                      tf.keras.layers.Dense(512, activation=tf.nn.relu),
                                      tf.keras.layers.Dropout(0.2),
                                      tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
model.compile(optimizer='adam',
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
model.compile(optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential(
    [tf.keras.layers.Flatten(),
     quantize.Quantize(tf.keras.layers.Dense(512, activation=tf.nn.relu)),
     tf.keras.layers.Dropout(0.2),
     quantize.Quantize(tf.keras.layers.Dense(10, activation=tf.nn.softmax))
    ]
)
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
Optimization

Quantization (post-training)

TensorFlow (estimator or Keras) → Saved Model → TF Lite Converter → TF Lite Model
Optimization
Quantization (post-training)

TensorFlow (estimator or Keras)
Saved Model + Calibration Data
TF Lite Converter
TF Lite Model
Optimization

Connection pruning

What does it mean?

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

Connection pruning

Benefits

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

Connection pruning

Coming soon

- Training with connection pruning in Keras-based API (compression benefits)

In the pipeline

- Inference support for sparse models (speed-ups on CPU and selected NPUs)
Optimization
Pruning results

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

![Graph showing MobileNet Top1 & Top5 Accuracy vs. Sparsity](image)
Model repository

Added new model repository

In depth sample applications & tutorials for:

- Image classification
- Object detection
- Pose estimation
- Segmentation
- Smart reply
TF Mobile Deprecated

- Provided 6+ months of notice
- Limiting developer support in favor of TensorFlow Lite
- Still available for training on Github
TensorFlow Lite for Microcontrollers
Smaller, cheaper & wider range of devices
What am I talking about?

Tiny models on tiny computers!

- Microcontrollers are everywhere
- Speech researchers were pioneers
- Models just tens of kilobytes
Here’s one I have in my pocket

Get ready for a live demo!

https://www.sparkfun.com/products/15170

384KB RAM, 1MB Flash, $15
Low single-digit milliwatt power usage
Days on a coin battery!
Why is this useful?

Running entirely on-device

Tiny constraints:

- It’s using a **20KB model**
- Runs using less than 100KB of RAM and 80KB of Flash
What is Coral?

- Coral is a platform for creating products with on-device ML acceleration.
- Our first products feature Google’s Edge TPU in SBC and USB accessory forms.
Edge TPU

A Google-designed ASIC that lets you run inference on-device:

- Very fast inference speed (object detection in less than 15ms)
- Enables greater data privacy
- No reliance on a network connection
- Runs inference with TensorFlow Lite

Enables unique workloads and new applications
Coral Dev Board

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>i.MX 8M SoC w/ Quad-core A53</td>
</tr>
<tr>
<td>GPU</td>
<td>Integrated GC7000 Lite GPU</td>
</tr>
<tr>
<td>TPU</td>
<td>Google Edge TPU</td>
</tr>
<tr>
<td>RAM Memory</td>
<td>1GB LPDDR4 RAM</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>8 GB eMMC</td>
</tr>
<tr>
<td>Security/Crypto</td>
<td>eMMC secure block for TrustZone MCHP ATECC608A Crypto Chip</td>
</tr>
<tr>
<td>Power</td>
<td>5V 3A via Type-C connector</td>
</tr>
<tr>
<td>Connectors</td>
<td>USB-C, RJ45, 3.5mm TRRS, HDMI</td>
</tr>
<tr>
<td>Supported OS</td>
<td>Mendel Linux (Debian derivative) Android</td>
</tr>
<tr>
<td>Supported ML</td>
<td>TensorFlow Lite</td>
</tr>
</tbody>
</table>
# Coral Accelerator

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPU</td>
<td>Google Edge TPU</td>
</tr>
<tr>
<td>Power</td>
<td>5V 3A via Type-C connector</td>
</tr>
<tr>
<td>Connectors</td>
<td>USB 3.1 (gen 1) via USB Type-C</td>
</tr>
<tr>
<td>Supported OS</td>
<td>Debian 6.0 or higher</td>
</tr>
<tr>
<td></td>
<td>Other Debian Derivatives</td>
</tr>
<tr>
<td>Supported Architectures</td>
<td>x86_64</td>
</tr>
<tr>
<td></td>
<td>ARMv8</td>
</tr>
<tr>
<td>Supported ML</td>
<td>TensorFlow Lite</td>
</tr>
</tbody>
</table>
These actually exist!

They're available now at coral.withgoogle.com
Get it. Try it.

**Code:** github.com/tensorflow/tensorflow

**Docs:** tensorflow.org/lite/

**Discuss:** tflite@tensorflow.org mailing list
Deep Learning MeetUp Group

The Group:

● > 3,500 members

The Meetings:

● Next = 16-April, hosted at Google
  ○ Something for Beginners
  ○ Something from the Bleeding Edge
  ○ Lightning Talks
Deep Learning JumpStart Workshop

This Saturday + (Tues & Thurs evening next week)

- Hands-on with real model code
- Build your own Project

Action points:

- Cost is heavily subsidised for SC/PR
Advanced Deep Learning Courses

Module #1: JumpStart (see previous slide)

Each 'module' will include:

● In-depth instruction, by practitioners
● Individual Projects
● 70%-100% funding via IMDA for SG/PR

Action points:

Red Dragon AI: Intern Hunt

Opportunity to do Deep Learning “all day”

Key Features:
- Work on something cutting-edge (+ publish!)
- Location: Singapore (SG/PR FTW) and/or Remote

Action points:
- Need to coordinate timing...
- Contact Martin or Sam via LinkedIn