In [ ]:

```
#@title Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

Image classification with TensorFlow Lite Model Maker

View on TensorFlow.org | Run in Google Colab | View source on GitHub | Download notebook | See TF Hub model

Prerequisites

To run this example, we first need to install several required packages, including Model Maker package that in GitHub repo.

In [1]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [2]:

```
!pip install -q tflite-model-maker-nightly
!pip install -q pycocotools
```

```
|████████████████████████████████| 593kB 8.5MB/s
|████████████████████████████████| 174kB 48.3MB/s
|████████████████████████████████| 112kB 57.4MB/s
|████████████████████████████████| 6.3MB 21.1MB/s
|████████████████████████████████| 122kB 55.4MB/s
|████████████████████████████████| 1.1MB 37.4MB/s
|████████████████████████████████| 92kB 11.7MB/s
|████████████████████████████████| 849kB 39.1MB/s
|████████████████████████████████| 1.2MB 17.8MB/s
|████████████████████████████████| 686kB 42.9MB/s
|████████████████████████████████| 645kB 48.8MB/s
|████████████████████████████████| 454.3MB 40kB/s
|████████████████████████████████| 194kB 57.2MB/s
|████████████████████████████████| 358kB 47.8MB/s
|████████████████████████████████| 686kB 42.1MB/s
|████████████████████████████████| 102kB 13.6MB/s
|████████████████████████████████| 38.2MB 77kB/s
|████████████████████████████████| 4.0MB 41.3MB/s
|████████████████████████████████| 471kB 45.0MB/s
|████████████████████████████████| 1.2MB 44.5MB/s
|████████████████████████████████| 6.0MB 19.2MB/s
|████████████████████████████████| 4.0MB 42.6MB/s
|████████████████████████████████| 4.9MB 37.1MB/s
```

```
  Building wheel for fire (setup.py) ... done
  Building wheel for py-cpuinfo (setup.py) ... done
```

Import the required packages.

In [1]:

```
import os

import numpy as np

import tensorflow as tf

from tflite_model_maker import configs
from tflite_model_maker import ExportFormat
from tflite_model_maker import image_classifier
from tflite_model_maker import ImageClassifierDataLoader
from tflite_model_maker import model_spec

import matplotlib.pyplot as plt
```

Simple End-to-End Example

If you prefer not to upload your images to the cloud, you could try to run the library locally following the guide in GitHub.

In [2]:

```
data = ImageClassifierDataLoader.from_folder("C:/Users/Joelp/OneDrive/Imagens/sement
train_data, val_data = data.split(0.7)
```

```
INFO:tensorflow:Load image with size: 40372, num_label: 43, labels: Acacia Mangium,
Acathospermum Hispidum, Ageratum Conyzoides, Albizia Hassleri Albizia, Allamanda Bla
nchetii, Allium Cepa, Amburana, Anacardium Humile, Apeiba Tibourbou, Arachis hypoge
a, Araucaria, Artocarpus, Aspidosperma macrocarpa, Avehoa carambola, Avena sativa, A
verrhoa Bilimbi, Begonia Angularis, Bertoletia Excelsa, Beta Vulgaris, Bixa Orellan
a, Brassica Oleracea, Brosimum Gaudichaudii, Caesalpinea Pucherina, Cajanjus Cajan,
Calistemum, Canavalia Ensiformes, Capsicum Annuum, Carica papaya, Cariniana Legalis,
Caryocar Brasiliense, Cassia Grandis, Cedrela Fissilis, Citrus, Cucumis Melo L, Cucu
rbita, Familia Annona, Malus Domestica, Passiflora edulis, Phaseolus vulgaris Pinto
Group, Spondias mombin, Tamarindus indica, Vigna unguiculata, Zea Mays.
```

In [3]:


```
INFO:tensorflow:Load image with size: 40372, num_label: 43, labels: Acacia Mangium,
Acathospermum Hispidum, Ageratum Conyzoides, Albizia Hassleri Albizia, Allamanda Bla
nchetii, Allium Cepa, Amburana, Anacardium Humile, Apeiba Tibourbou, Arachis hypoge
a, Araucaria, Artocarpus, Aspidosperma macrocarpa, Avehoa carambola, Avena sativa, A
verrhoa Bilimbi, Begonia Angularis, Bertoletia Excelsa, Beta Vulgaris, Bixa Orellan
a, Brassica Oleracea, Brosimum Gaudichaudii, Caesalpinea Pucherina, Cajanjus Cajan,
Calistemum, Canavalia Ensiformes, Capsicum Annuum, Carica papaya, Cariniana Legalis,
Caryocar Brasiliense, Cassia Grandis, Cedrela Fissilis, Citrus, Cucumis Melo L, Cucu
rbita, Familia Annona, Malus Domestica, Passiflora edulis, Phaseolus vulgaris Pinto
Group, Spondias mombin, Tamarindus indica, Vigna unguiculata, Zea Mays.
```


Step 2. Customize the TensorFlow model.

In [4]:

```
model = image_classifier.create(train_data, validation_data=val_data,  epochs=20) #,
```

```
INFO:tensorflow:Retraining the models...
WARNING:tensorflow:Please add `keras.layers.InputLayer` instead of `keras.Input` to
Sequential model. `keras.Input` is intended to be used by Functional model.
WARNING:tensorflow:Please add `keras.layers.InputLayer` instead of `keras.Input` to
Sequential model. `keras.Input` is intended to be used by Functional model.
Model: "sequential"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
hub_keras_layer_v1v2 (HubKer (None, 1280)              3413024
_________________________________________________________________
dropout (Dropout)            (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 43)                55083
=================================================================
Total params: 3,468,107
Trainable params: 55,083
Non-trainable params: 3,413,024
_________________________________________________________________
None
c:\users\joelp\appdata\local\programs\python\python37\lib\site-packages\tensorflow\p
ython\keras\optimizer_v2\optimizer_v2.py:375: UserWarning: The `lr` argument is depr
ecated, use `learning_rate` instead.
  "The `lr` argument is deprecated, use `learning_rate` instead.")
Epoch 1/20
883/883 [==============================] - 959s 1s/step - loss: 1.3496 - accuracy:
0.8523 - val_loss: 0.9392 - val_accuracy: 0.9776
Epoch 2/20
883/883 [==============================] - 862s 976ms/step - loss: 0.9338 - accurac
y: 0.9785 - val_loss: 0.8636 - val_accuracy: 0.9907
Epoch 3/20
883/883 [==============================] - 876s 992ms/step - loss: 0.8844 - accurac
y: 0.9884 - val_loss: 0.8339 - val_accuracy: 0.9950
Epoch 4/20
883/883 [==============================] - 896s 1s/step - loss: 0.8626 - accuracy:
0.9919 - val_loss: 0.8183 - val_accuracy: 0.9969
Epoch 5/20
883/883 [==============================] - 904s 1s/step - loss: 0.8484 - accuracy:
0.9944 - val_loss: 0.8085 - val_accuracy: 0.9979
Epoch 6/20
883/883 [==============================] - 906s 1s/step - loss: 0.8393 - accuracy:
0.9954 - val_loss: 0.8017 - val_accuracy: 0.9979
Epoch 7/20
883/883 [==============================] - 941s 1s/step - loss: 0.8320 - accuracy:
0.9963 - val_loss: 0.7964 - val_accuracy: 0.9986
Epoch 8/20
883/883 [==============================] - 914s 1s/step - loss: 0.8263 - accuracy:
0.9975 - val_loss: 0.7923 - val_accuracy: 0.9988
Epoch 9/20
883/883 [==============================] - 987s 1s/step - loss: 0.8220 - accuracy:
0.9973 - val_loss: 0.7888 - val_accuracy: 0.9990
Epoch 10/20
883/883 [==============================] - 976s 1s/step - loss: 0.8183 - accuracy:
0.9982 - val_loss: 0.7866 - val_accuracy: 0.9991
Epoch 11/20
883/883 [==============================] - 917s 1s/step - loss: 0.8156 - accuracy:
0.9982 - val_loss: 0.7844 - val_accuracy: 0.9993
Epoch 12/20
883/883 [==============================] - 961s 1s/step - loss: 0.8131 - accuracy:
0.9983 - val_loss: 0.7817 - val_accuracy: 0.9993
Epoch 13/20
883/883 [==============================] - 928s 1s/step - loss: 0.8105 - accuracy:
0.9985 - val_loss: 0.7805 - val_accuracy: 0.9994
Epoch 14/20
883/883 [==============================] - 936s 1s/step - loss: 0.8085 - accuracy:
0.9985 - val_loss: 0.7787 - val_accuracy: 0.9993
Epoch 15/20
883/883 [==============================] - 931s 1s/step - loss: 0.8075 - accuracy:
0.9983 - val_loss: 0.7781 - val_accuracy: 0.9993
Epoch 16/20
883/883 [==============================] - 898s 1s/step - loss: 0.8055 - accuracy:
0.9987 - val_loss: 0.7763 - val_accuracy: 0.9993
Epoch 17/20
883/883 [==============================] - 931s 1s/step - loss: 0.8039 - accuracy:
0.9989 - val_loss: 0.7753 - val_accuracy: 0.9996
Epoch 18/20
```

```
883/883 [==============================] - 929s 1s/step - loss: 0.8024 - accuracy:
0.9987 - val_loss: 0.7745 - val_accuracy: 0.9997
Epoch 19/20
883/883 [==============================] - 1063s 1s/step - loss: 0.8019 - accuracy:
0.9988 - val_loss: 0.7734 - val_accuracy: 0.9998
Epoch 20/20
883/883 [==============================] - 989s 1s/step - loss: 0.8010 - accuracy:
0.9991 - val_loss: 0.7735 - val_accuracy: 0.9998
```

Testar o modelo

In [11]:

```
test_data = ImageClassifierDataLoader.from_folder("C:/Users/Joelp/OneDrive/Imagens/s
test_data = data.split(0.9)
```

```
INFO:tensorflow:Load image with size: 1280, num_label: 44, labels: Acacia Mangium, A
cathospermum Hispidum, Ageratum Conyzoides, Albizia Hassleri Albizia, Allamanda Blan
chetii, Allium Cepa, Amburana, Anacardium Humile, Apeiba Tibourbou, Arachis hypogea,
Araucaria, Artocarpus, Aspidosperma macrocarpa, Avehoa carambola, Avena sativa, Aver
rhoa Bilimbi, Begonia Angularis, Bertoletia Excelsa, Beta Vulgaris, Bixa Orellana, B
rassica Oleracea, Brosimum Gaudichaudii, Caesalpinea Pucherina, Cajanjus Cajan, Cali
stemum, Canavalia Ensiformes, Capsicum Annuum, Carica papaya, Cariniana Legalis, Car
yocar Brasiliense, Cassia Grandis, Cedrela Fissilis, Citrus, Cucumis Melo L, Cucurbi
ta, Familia Annona, Malus Domestica, Não Classificado, Passiflora edulis, Phaseolus
vulgaris Pinto Group, Spondias mombin, Tamarindus indica, Vigna unguiculata, Zea May
s.
```


```
INFO:tensorflow:Load image with size: 1280, num_label: 44, labels: Acacia Mangium, A
cathospermum Hispidum, Ageratum Conyzoides, Albizia Hassleri Albizia, Allamanda Blan
chetii, Allium Cepa, Amburana, Anacardium Humile, Apeiba Tibourbou, Arachis hypogea,
Araucaria, Artocarpus, Aspidosperma macrocarpa, Avehoa carambola, Avena sativa, Aver
rhoa Bilimbi, Begonia Angularis, Bertoletia Excelsa, Beta Vulgaris, Bixa Orellana, B
rassica Oleracea, Brosimum Gaudichaudii, Caesalpinea Pucherina, Cajanjus Cajan, Cali
stemum, Canavalia Ensiformes, Capsicum Annuum, Carica papaya, Cariniana Legalis, Car
yocar Brasiliense, Cassia Grandis, Cedrela Fissilis, Citrus, Cucumis Melo L, Cucurbi
ta, Familia Annona, Malus Domestica, Não Classificado, Passiflora edulis, Phaseolus
vulgaris Pinto Group, Spondias mombin, Tamarindus indica, Vigna unguiculata, Zea May
s.
```


In [12]:

```
loss, accuracy = model.evaluate(test_data[0])
```

```
1136/1136 [==============================] - 846s 745ms/step - loss: 0.7662 - accura
cy: 0.9998
```

Step 4. Export to TensorFlow Lite model.

Here, we export TensorFlow Lite model with metadata which provides a standard for model descriptions. The label file is embedded in metadata.

You could download it in the left sidebar same as the uploading part for your own use.

Step 4: Export to TensorFlow Lite Model

Convert the existing model to TensorFlow Lite model format with metadata. The default TFLite filename is `model.tflite`.

See example applications and guides of image classification for more details about how to integrate the TensorFlow Lite model into mobile apps.

The allowed export formats can be one or a list of the following:

- `ExportFormat.TFLITE`
- `ExportFormat.LABEL`
- `ExportFormat.SAVED_MODEL`

By default, it just exports TensorFlow Lite model with metadata. You can also selectively export different files. For instance, exporting only the label file as follows:

In [21]:
```
import os
os.mkdir("modelo-sementes-24-05-2021/")
```

In [22]:
```
model.export(export_dir='modelo-sementes-24-05-2021/', export_format=ExportFormat.TF
```

```
INFO:tensorflow:Assets written to: C:\Users\Joelp\AppData\Local\Temp\tmpcumf2fct\ass
ets
INFO:tensorflow:Assets written to: C:\Users\Joelp\AppData\Local\Temp\tmpcumf2fct\ass
ets
WARNING:absl:For model inputs containing unsupported operations which cannot be quan
tized, the `inference_input_type` attribute will default to the original type.
INFO:tensorflow:Label file is inside the TFLite model with metadata.
INFO:tensorflow:Label file is inside the TFLite model with metadata.
INFO:tensorflow:Saving labels in C:\Users\Joelp\AppData\Local\Temp\tmppqx6hgff\label
s.txt
INFO:tensorflow:Saving labels in C:\Users\Joelp\AppData\Local\Temp\tmppqx6hgff\label
s.txt
INFO:tensorflow:TensorFlow Lite model exported successfully: modelo-24-05-2021/mode
l.tflite
INFO:tensorflow:TensorFlow Lite model exported successfully: modelo-24-05-2021/mode
l.tflite
```

In [ ]:
```
model.evaluate_tflite('modelo-sementes-24-05-2021/model.tflite', test_data[0])
```

Post-training quantization on the TensorFLow Lite model

Post-training quantization is a conversion technique that can reduce model size and inference latency, while also improving CPU and hardware accelerator latency, with little degradation in model accuracy. Thus, it's widely used to optimize the model.

Model Maker supports multiple post-training quantization options. Let's take full integer quantization as an instance. First, define the quantization config to enforce full integer quantization for all ops including the input and output. The input type and output type are `uint8` by default. You may also change them to other types like `int8` by setting `inference_input_type` and `inference_output_type` in config.

In [29]:
```
config = config = configs.QuantizationConfig().representative_data(representative_da
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-29-9c065dfc93dd> in <module>
----> 1 config = config = configs.QuantizationConfig().representative_data(represent
ative_data=test_data[0], is_integer_only=True)

TypeError: 'NoneType' object is not callable
```

Then we export TensorFlow Lite model with such configuration.

In [27]:
```
model.export(export_dir='modelo-sementes-24-05-2021/', tflite_filename='quantizedIN8
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
```

```
<ipython-input-27-df57cb635517> in <module>
----> 1 model.export(export_dir='modelo-sementes-24-05-2021/', tflite_filename='quantizedIN8.tflite', quantization_config=configs.QuantizationConfig.get_converter_with_quantization(representative_data=test_data[0], is_integer_only=True))

TypeError: get_converter_with_quantization() missing 2 required positional arguments: 'self' and 'converter'
```

In Colab, you can download the model named `model_quant.tflite` from the left sidebar, same as the uploading part mentioned above.

Change the model

Change to the model that's supported in this library.

This library supports EfficientNet-Lite models, MobileNetV2, ResNet50 by now. EfficientNet-Lite are a family of image classification models that could achieve state-of-art accuracy and suitable for Edge devices. The default model is EfficientNet-Lite0.

We could switch model to MobileNetV2 by just setting parameter `model_spec` to `mobilenet_v2_spec` in `create` method.

In [ ]:
```
model = image_classifier.create(train_data, model_spec=model_spec.mobilenet_v2_spec,
```

Evaluate the newly retrained MobileNetV2 model to see the accuracy and loss in testing data.

In [ ]:
```
loss, accuracy = model.evaluate(test_data)
```

Change to the model in TensorFlow Hub

Moreover, we could also switch to other new models that inputs an image and outputs a feature vector with TensorFlow Hub format.

As Inception V3 model as an example, we could define `inception_v3_spec` which is an object of `ImageModelSpec` and contains the specification of the Inception V3 model.

We need to specify the model name `name`, the url of the TensorFlow Hub model `uri`. Meanwhile, the default value of `input_image_shape` is `[224, 224]`. We need to change it to `[299, 299]` for Inception V3 model.

In [ ]:
```
inception_v3_spec = model_spec.ImageModelSpec(
    uri='https://tfhub.dev/google/imagenet/inception_v3/feature_vector/1')
inception_v3_spec.input_image_shape = [299, 299]
```

Then, by setting parameter `model_spec` to `inception_v3_spec` in `create` method, we could retrain the Inception V3 model.

The remaining steps are exactly same and we could get a customized InceptionV3 TensorFlow Lite model in the end.

Change your own custom model

If we'd like to use the custom model that's not in TensorFlow Hub, we should create and export ModelSpec in TensorFlow Hub.

Then start to define `ImageModelSpec` object like the process above.

Change the training hyperparameters

We could also change the training hyperparameters like `epochs`, `dropout_rate` and `batch_size` that could affect the model accuracy. The model parameters you can adjust are:

- `epochs` : more epochs could achieve better accuracy until it converges but training for too many epochs may lead to overfitting.
- `dropout_rate` : The rate for dropout, avoid overfitting. None by default.
- `batch_size` : number of samples to use in one training step. None by default.
- `validation_data` : Validation data. If None, skips validation process. None by default.
- `train_whole_model` : If true, the Hub module is trained together with the classification layer on top. Otherwise, only train the top classification layer. None by default.
- `learning_rate` : Base learning rate. None by default.
- `momentum` : a Python float forwarded to the optimizer. Only used when `use_hub_library` is True. None by default.
- `shuffle` : Boolean, whether the data should be shuffled. False by default.
- `use_augmentation` : Boolean, use data augmentation for preprocessing. False by default.
- `use_hub_library` : Boolean, use `make_image_classifier_lib` from tensorflow hub to retrain the model. This training pipeline could achieve better performance for complicated dataset with many categories. True by default.
- `warmup_steps` : Number of warmup steps for warmup schedule on learning rate. If None, the default warmup_steps is used which is the total training steps in two epochs. Only used when `use_hub_library` is False. None by default.
- `model_dir` : Optional, the location of the model checkpoint files. Only used when `use_hub_library` is False. None by default.

Parameters which are None by default like `epochs` will get the concrete default parameters in make_image_classifier_lib from TensorFlow Hub library or train_image_classifier_lib.

For example, we could train with more epochs.

In [ ]:

```
model = image_classifier.create(train_data, validation_data=validation_data, epochs=
```

Evaluate the newly retrained model with 10 training epochs.

In [ ]:

```
loss, accuracy = model.evaluate(test_data)
```