TensorFlow Lite - Running Machine Learning Models on Android&Embedded devices

Deploy machine learning models on mobile and IoT devices

Dinakar Maurya

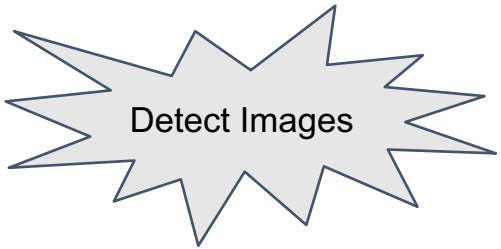
Table of Contents

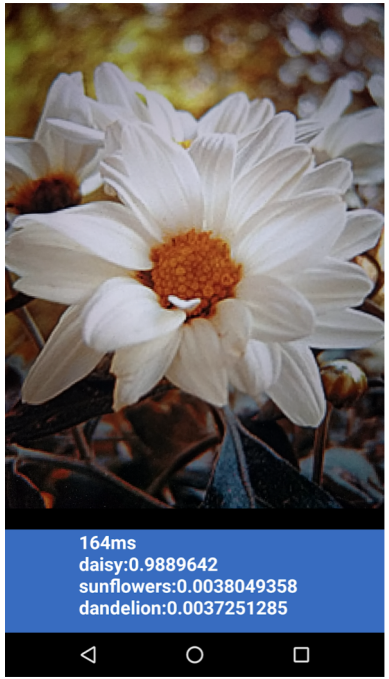
What, Why, How

Build your first TensorFlow Lite app

What you will build

A simple android camera app that runs a TensorFlow image recognition program to identify flowers.





Scope - Optimized models for common mobile and edge use cases

- Image classification

Identify hundreds of objects, including people, activities, animals, plants, and places.

- Object detection

Detect multiple objects with bounding boxes. Yes, dogs and cats too.

- Smart reply

Generate reply suggestions to input conversational chat messages.

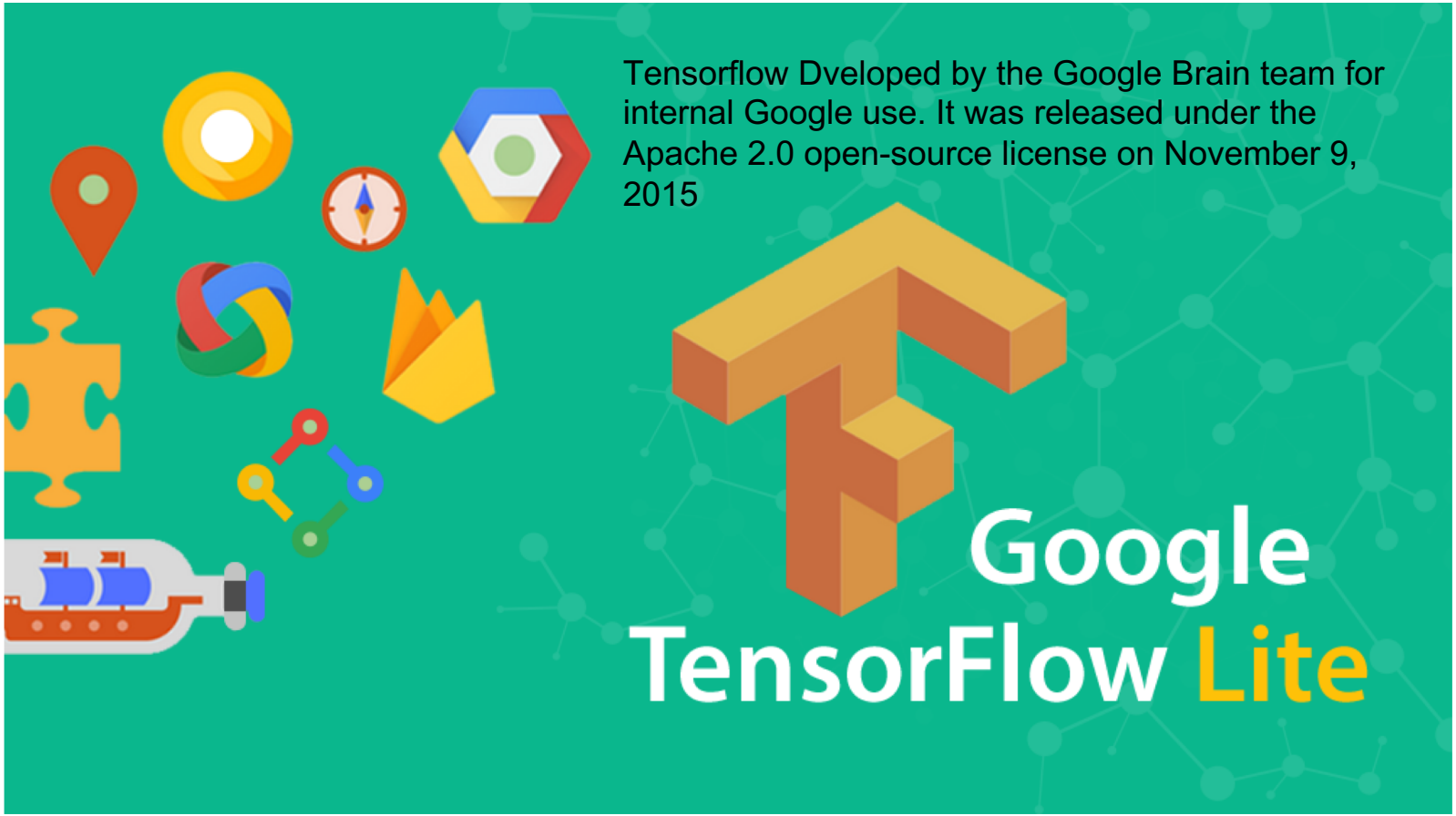
- Pose estimation

Estimate poses for single or multiple people. Imagine the possibilities, including stick figure dance parties.

- Segmentation

Pinpoint the shape of objects with strict localization accuracy and semantic labels. Trained with people, places, animals, and more.

What is TensorFlow & TensorFlow Lite



What is TensorFlow & TensorFlow Lite

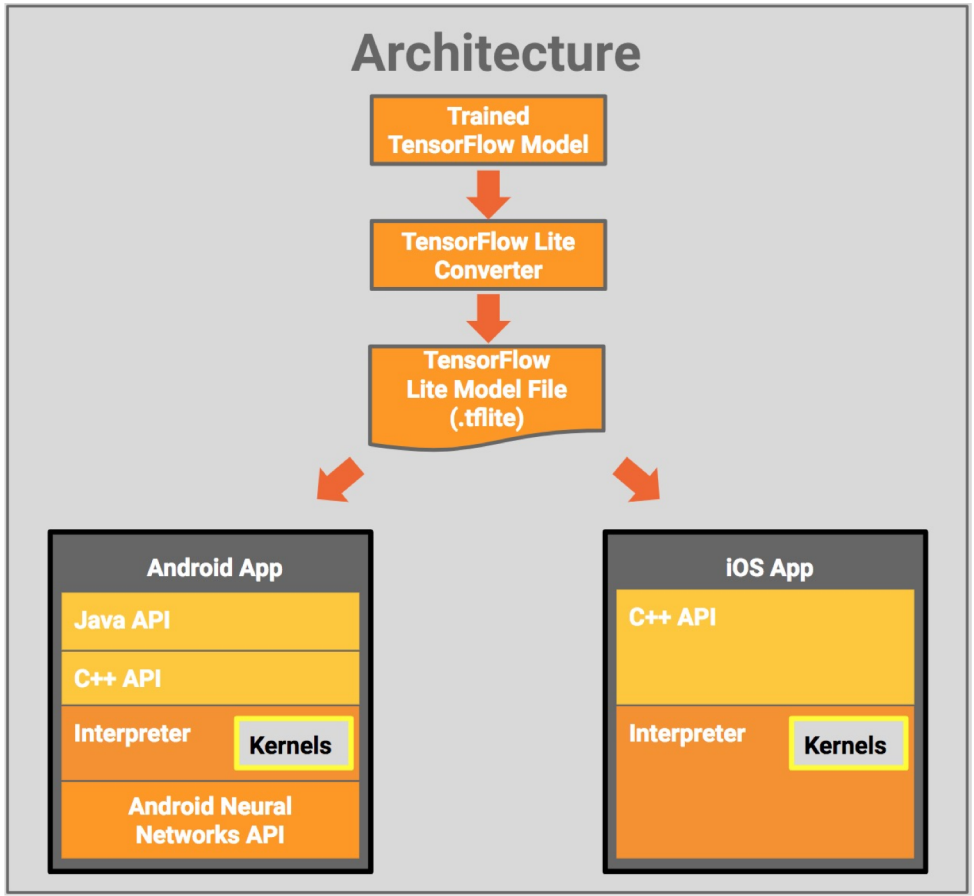
TensorFlow

- TensorFlow is an open-source software library for dataflow and differentiable programming across a range of tasks.
- It is a symbolic math library, and is also used for machine learning applications such as neural networks.
- TensorFlow is a multipurpose machine learning framework.
- TensorFlow can be used anywhere from training huge models across clusters in the cloud, to running models locally on an embedded system like your phone.

TensorFlow Lite

- TensorFlow Lite is for mobile and embedded devices.
- TensorFlow Lite is the official solution for running machine learning models on mobile and embedded devices.
- It enables on-device machine learning inference with low latency and a small binary size on Android, iOS, and other operating systems.

TensorFlow Lite Architecture



TensorFlow Lite Users

- Use AI in Mobile Devices and Embedded Devices.
- Enables on-device machine learning inference with low latency and a small binary size on Android, iOS, and other operating systems.
- Some of tensorFlow Lite users -

TensorFlow Lite example apps

- Gesture recognition
  - Train a neural network to recognize gestures caught on your webcam using TensorFlow.js,
  - then use TensorFlow Lite to convert the model to run inference on your device.
- Image classification
  - Test an image classification solution with a pre-trained model that can recognize 1000 different types of items from input frames on a mobile camera.
- Object detection
  - Explore an app using a pre-trained model that draws and labels bounding boxes around 1000 different recognizable objects from input frames on a mobile camera.
- Speech recognition
  - Explore an app that uses a microphone to spot keywords and return a probability score for the words spoken.

Benifits

On-device ML inference is difficult because of the many constraints—TensorFlow Lite can solve these:

- Performance
  a. TF Lite is fast with no noticeable accuracy loss.
- Portability
  a. Android, iOS, and more specialized IoT devices.
- Low latency
  a. Optimized float- and fixed-point CPU kernels.
- Acceleration
  a. Integration with GPU and internal/external accelerators.
  b. TensorFlow Lite also supports hardware acceleration with the Android Neural Networks API.
- Small model size
  a. Controlled dependencies, quantization and op registration.
- Tooling
  a. Conversion, compression, benchmarking, power-consumption, and more.

How it works

Pick a model

Pick a new model or retrain an existing one.

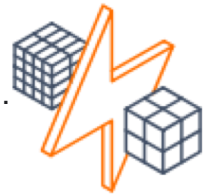
Convert

Convert a TensorFlow model into a compressed flat buffer with the TensorFlow Lite Converter.

Deploy

Take the compressed `.tflite` file and load it into a mobile or embedded device.

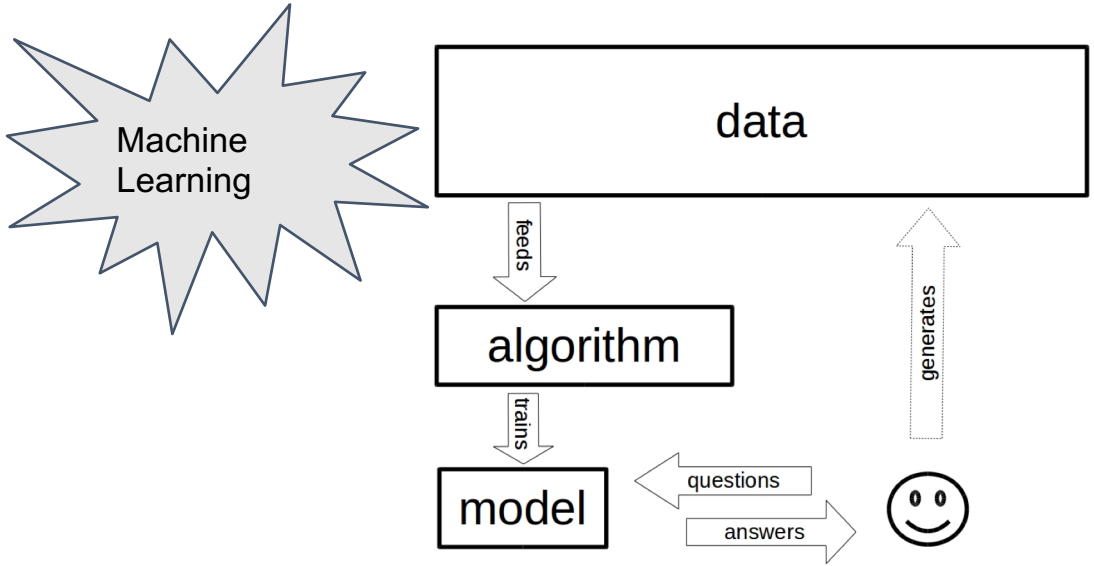
Optimize



Quantize by converting 32-bit floats to more efficient 8-bit integers or run on GPU.

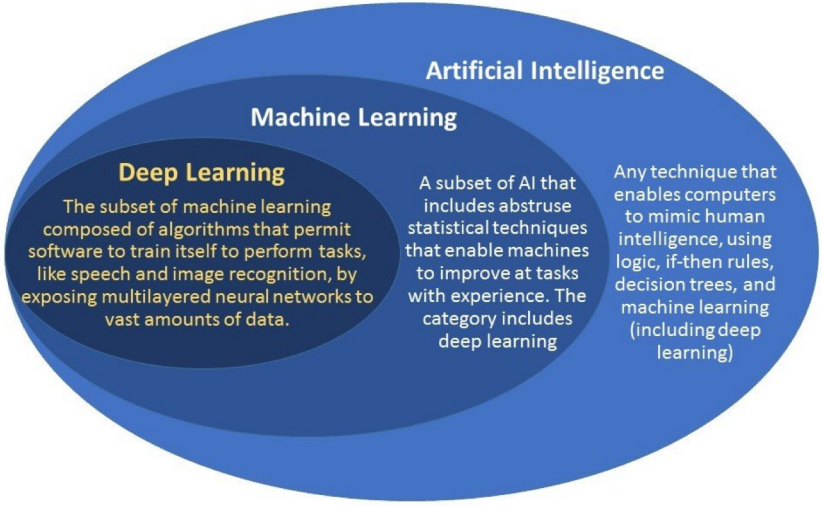
Some Basic Terms

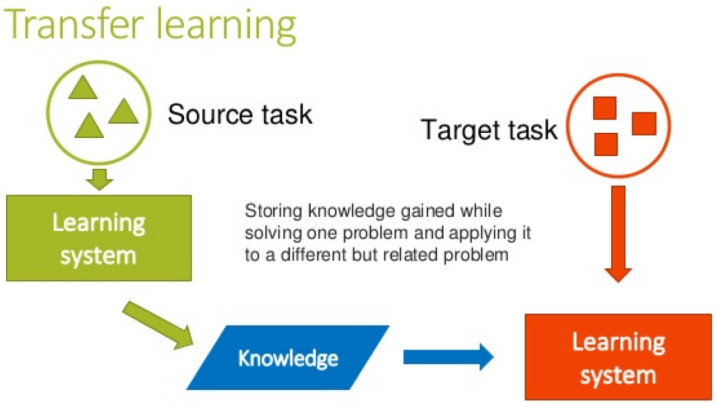
Before Starting - Some basic terms

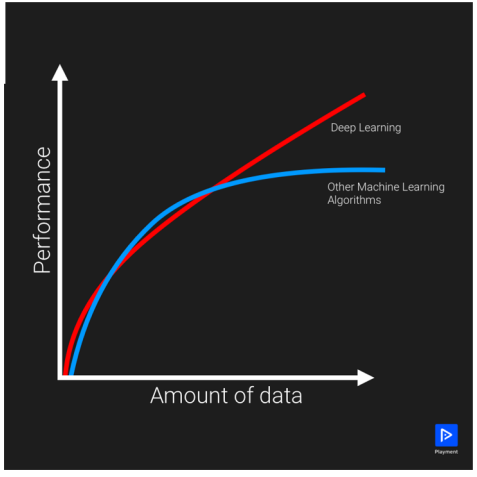


Machine Learning Classifiers

Before Starting - Some basic terms







Before Starting - Some basic terms

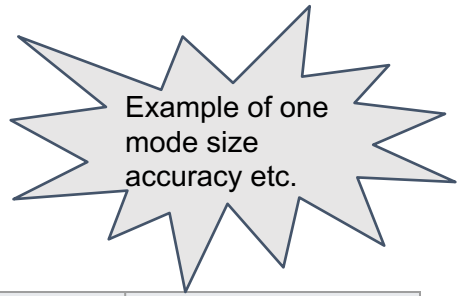
- Transfer learning
  - Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.
  - For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks.

- Pre-trained model(https://www.tensorflow.org/lite/guide/hosted_models)
  - A pre-trained model is a model that was trained on a large benchmark dataset to solve a problem similar to the one that we want to solve.
  - Models-
    - Inception v3 - 1000 classes - image dataset
      - It is trained for the ImageNet Large Visual Recognition Challenge using the data from 2012. This is a standard task in computer vision, where models try to classify entire images into 1000 classes.
    - MobileNet - A class of convolutional neural network designed by researchers at Google. They are coined “mobile-first” in that they’re architected from the ground up to be resource-friendly and run quickly, right on your phone.

- PILLOW- Python Imaging Library is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

Deep Learning Models - pre trained

- Object detection
  - - coco_ssd_mobilenet_v1_1.0_quant_2018_06_29 - Recognize 80 different classes of objects
- Pose estimation
  - -multi_person_mobilenet_v1_075_float - 16 poses (nose -1, leftEye - 2, rightEye - 3 etc.)
- Image segmentation
  - - deeplabv3_257_mv_gpu
- Smart reply
  - - smartreply_1.0_2017_11_01



Model name	Paper and model	Model size	Top-1 accuracy	Top-5 accuracy	TF Lite performance
Inception_V3_quant	paper,tflite&pb	23 Mb	77.5%	93.7%	637 ms

Before Starting - Some basic terms

1. Model
    a. A machine learning model can be a mathematical representation of a real-world process.
    b. The output of the training process is a machine learning model which you can then use to make predictions.

1. Train
    a. The training data is used to make sure the machine recognizes patterns in the data.
    b. The cross-validation data is used to ensure better accuracy and efficiency of the algorithm used to train the machine.
    c. And the test data is used to see how well the machine can predict new answers based on its training.

1. Classifier(Machine Learning Classifiers)
    a. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories.
    b. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).
        i. Linear Classifiers: Logistic Regression, Naive Bayes Classifier.
        ii. Support Vector Machines.
        iii. Decision Trees.
        iv. Boosted Trees.
        v. Random Forest.
        vi. Neural Networks.
        vii. Nearest Neighbor.

1. Protocol Buffers(.pb) also known as protobuf, is Google's language-neutral, platform-neutral, extensible mechanism that is used to serialize structured data.

1. .tflite(Tensorflow Lite FlatBuffer File) -
    a. Converted form of machine learning trained model.
    b. This(.tflite file) gets loaded in to a mobile interpreter.

Python Setup & Generate Tensorflow lite file(.lite)

We build .lite and .txt files

- .lite - describing the models
- .txt file - describing the lables at the models trained for.

Example of .txt label file -

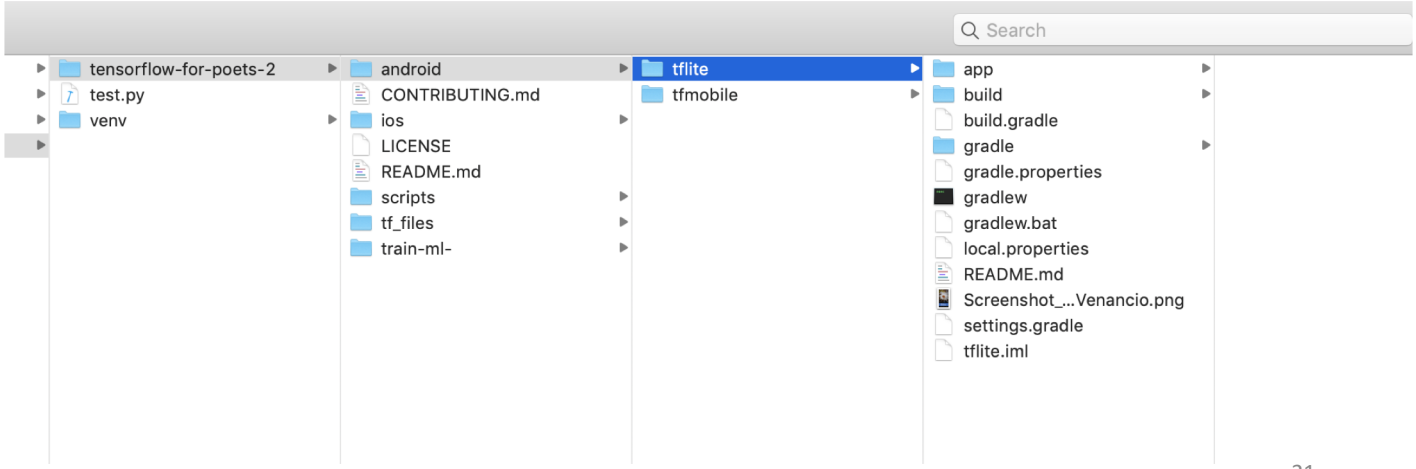
camellia
canna lily
canterbury bells
cape flower
carnation
cautleya spicata
clematis
colt s foot
columbine
common dandelion
corn poppy
cyclamen
daffodil
daisy
dandelion
desert rose

daisy - example of label

Setup Tensorflow

Steps-

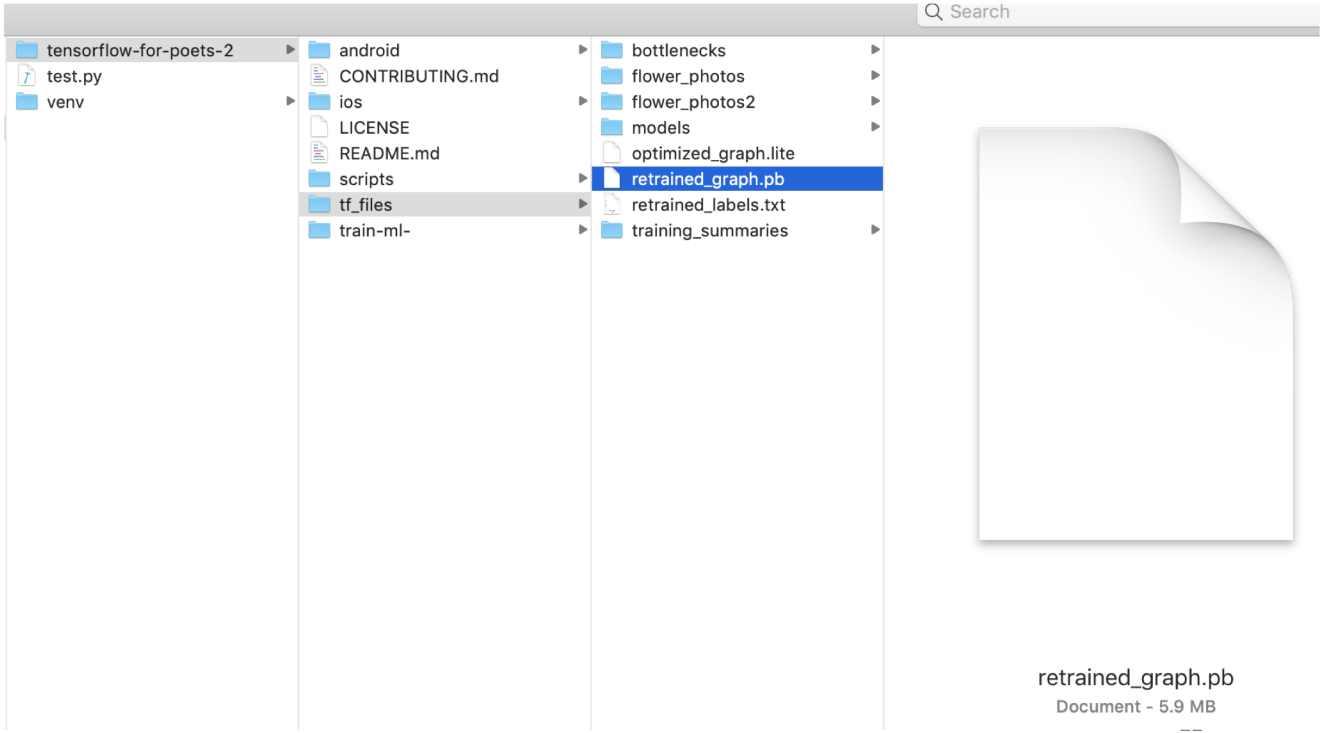
1. pip install --upgrade "tensorflow==1.7.*"
2. pip install PILLOW
3. git clone https://github.com/googlecodelabs/tensorflow-for-poets-2
4. cd tensorflow-for-poets-2
5. git checkout end_of_first_codelab
6. ls tf_files



Test the Model

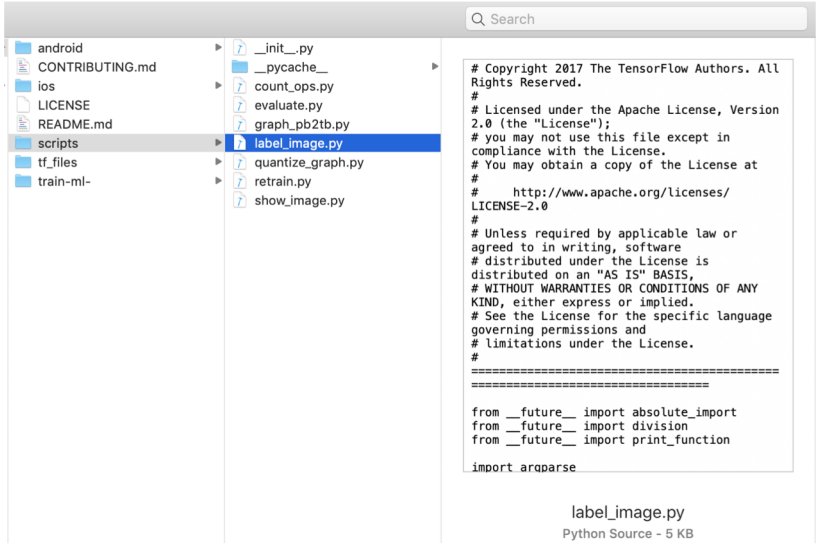
The scripts/ directory contains a simple command line script, label_image.py to test the network.
Now we'll test label_image.py on this picture of some daisies:

flower_photos/daisy/3475870145_685a19116d.jpg



Test the Model

```
python -m scripts.label_image ¥
  --graph=tf_files/retrained_graph.pb  ¥
--image=tf_files/flower_photos/daisy/3475870145_685a19116d.jpg
```



The script will print the probability the model has assigned to each flower type.
Something like this:
Evaluation time (1-image): 0.140s
daisy 0.7361
dandelion 0.242222
tulips 0.0185161
roses 0.0031544
sunflowers 8.00981e-06

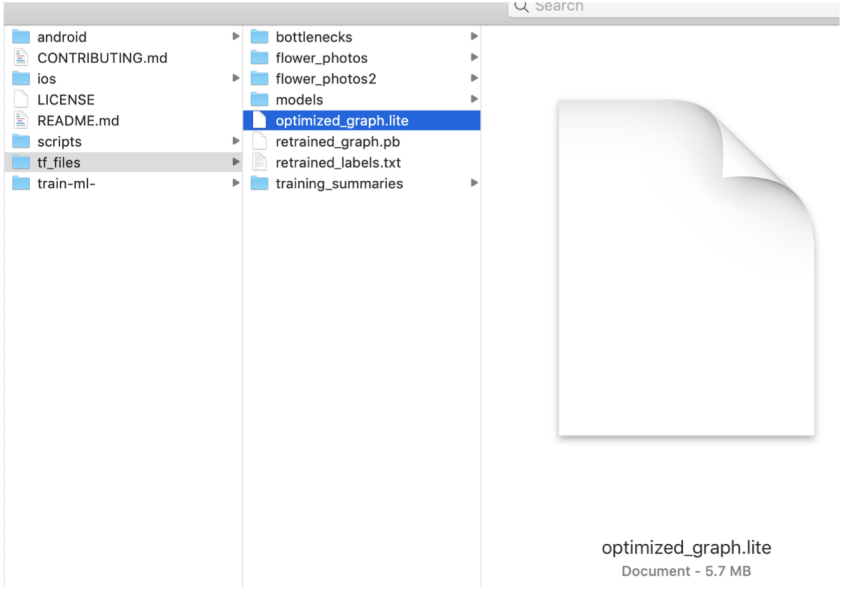
This should hopefully produce a sensible top label for your example.
You'll be using this command to make sure you're still getting sensible results
as you do further processing on the model file to prepare it for use in a mobile app.

Convert to .lite - Using the TFLite converter

Convert to model to TFLite format

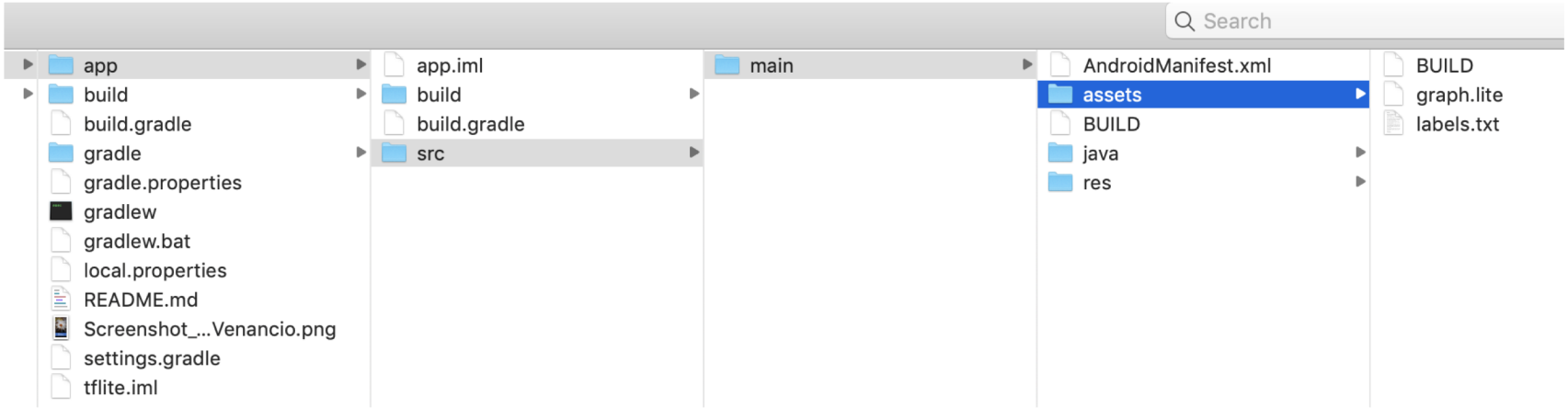
The TensorFlow Lite Converter takes a TensorFlow graph file and creates a graph file used by the TensorFlow Lite interpreter.

```
IMAGE_SIZE=224
tflite_convert ¥
  --graph_def_file=tf_files/retrained_graph.pb ¥
  --output_file=tf_files/optimized_graph.lite ¥
  --input_format=TENSORFLOW_GRAPHDEF ¥
  --output_format=TFLITE ¥
  --input_shape=1,${IMAGE_SIZE},${IMAGE_SIZE},3 ¥
  --input_array=input ¥
  --output_array=final_result ¥
  --inference_type=FLOAT ¥
  --input_data_type=FLOAT
```



Add your model files to the project

cp tf_files/optimized_graph.lite android/tflite/app/src/main/assets/graph.lite
cp tf_files/retrained_labels.txt android/tflite/app/src/main/assets/labels.txt



Android Studio Setup

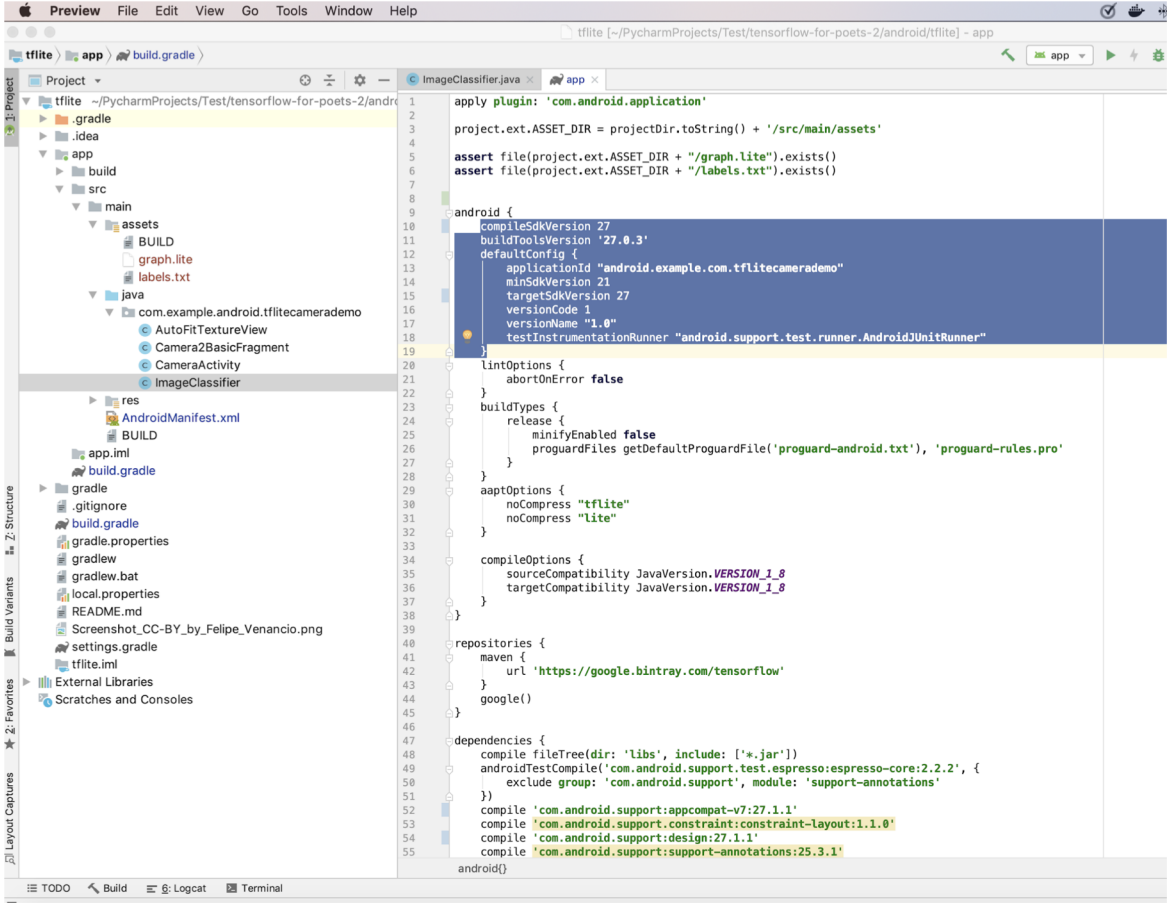
Setup the Android app

Assumptions-You already setup Android Studio IDE

Import tensorflow-for-poets-2/android/tflite

Build the application

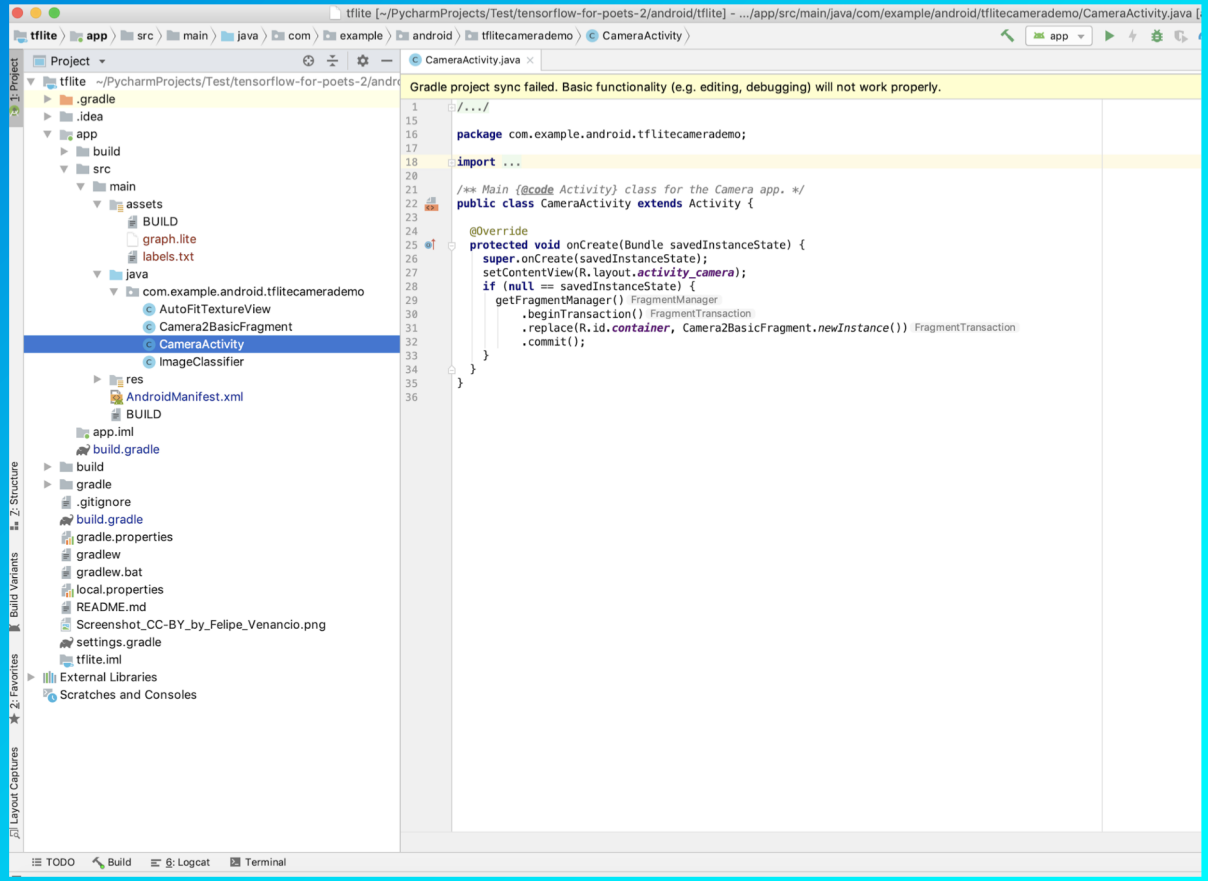
Setup the Android app - change the android tag configuration if required



Test the Android app - Running Machine Learning Models

- Connect USB cable on System and Run the app using Android Studio.
- This version of the app uses the standard MobileNet, pre-trained on the 1000 ImageNet categories.

Live Example / Running example with android studio



How does it work - Android Code

How does it work

- This app uses a pre-compiled TFLite Android Archive (AAR).
- This AAR is hosted on jcenter.
- The following lines in the module's build.gradle file include the newest version of the AAR, from the TensorFlow bintray maven repository, in the
- project.build.gradle - see image

```
repositories {
    maven {
        url 'https://google.bintray.com/tensorflow'
    }
}

dependencies {
    // ...
    compile 'org.tensorflow:tensorflow-lite:+'
}
```

- We use the following block, to instruct the Android Asset Packaging Tool that `.lite` or `.tflite` assets should not be compressed.
- This is important as the `.lite` file will be memory-mapped, and that will not work when the file is compressed.

```
android {
    aaptOptions {
        noCompress "tflite"
        noCompress "lite"
    }
}
```

How does it work - Using theTFLite Java API

- The code interfacing to the TFLite is all contained in ImageClassifier.java.

- The first block of interest is the constructor for the ImageClassifier:

```
ImageClassifier(Activity activity) throws IOException {
    tflite = new Interpreter(loadModelFile(activity));
    labelList = loadLabelList(activity);
    imgData =
        ByteBuffer.allocateDirect(
            4 * DIM_BATCH_SIZE * DIM_IMG_SIZE_X * DIM_IMG_SIZE_Y * DIM_PIXEL_SIZE);
    imgData.order(ByteOrder.nativeOrder());
    labelProbArray = new float[1][labelList.size()];
    Log.d(TAG, "Created a Tensorflow Lite Image Classifier.");
}
```

The following line creates the TFLite interpreter:
ImageClassifier.java

- This line instantiates a TFLite interpreter.
- The interpreter does the job of a tf.Session (for those familiar with TensorFlow, outside of TFLite).
- We pass the interpreter a MappedByteBuffer containing the model. The local function loadModelFile creates a MappedByteBuffer containing the activity's graph.lite asset file.

```
tflite = new Interpreter(loadModelFile(activity));
```

How does it work

- The following lines create the input data buffer:
- This byte buffer is sized to contain the image data once converted to float.
- The interpreter can accept float arrays directly as input, but the ByteBuffer is more efficient as it avoids extra copies in the interpreter.

```
imgData = ByteBuffer.allocateDirect(
    4 * DIM_BATCH_SIZE * DIM_IMG_SIZE_X * DIM_IMG_SIZE_Y * DIM_PIXEL_SIZE);
```

- The following lines load the label list and create the output buffer:
- The output buffer is a float array with one element for each label where the model will write the output probabilities.

```
labelList = loadLabelList(activity);
//...
labelProbArray = new float[1][labelList.size()];
```

How does it work - Run the model

- The second block of interest is the classifyFrame method.
- It takes a Bitmap as input, runs the model and returns the text to print in the app.
- This method does three things. First converts and copies the input Bitmap to the imgData ByteBuffer for input to the model.
- Then it calls the interpreter's run method, passing the input buffer and the output array as arguments.
- The interpreter sets the values in the output array to the probability calculated for each class.

```
String classifyFrame(Bitmap bitmap) {
 // ...
 convertBitmapToByteBuffer(bitmap);
 // ...
 tflite.run(imgData, labelProbArray);
 // ...
 String textToShow = printTopKLabels();
 // ...
}
```

```
Map.Entry<String, Float> label = sortedLabels.poll();
textToShow = String.format("\n%s: %4.2f",label.getKey(),label.getValue()) + textToShow;
```

References

- https://youtu.be/JnhW5tQ_7Vo
- https://www.tensorflow.org/lite
- https://www.tensorflow.org/lite/guide/android
- https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623
- https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e
- https://www.tensorflow.org/lite/models/object_detection/overview
- https://www.youtube.com/channel/UC0rqucBdTuFTjJiefW5t-IQ/playlists
- https://github.com/tensorflow/examples/tree/master/lite/examples

Questions!