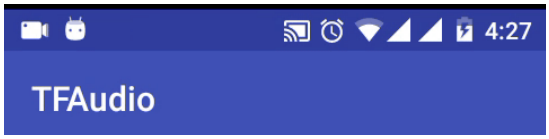
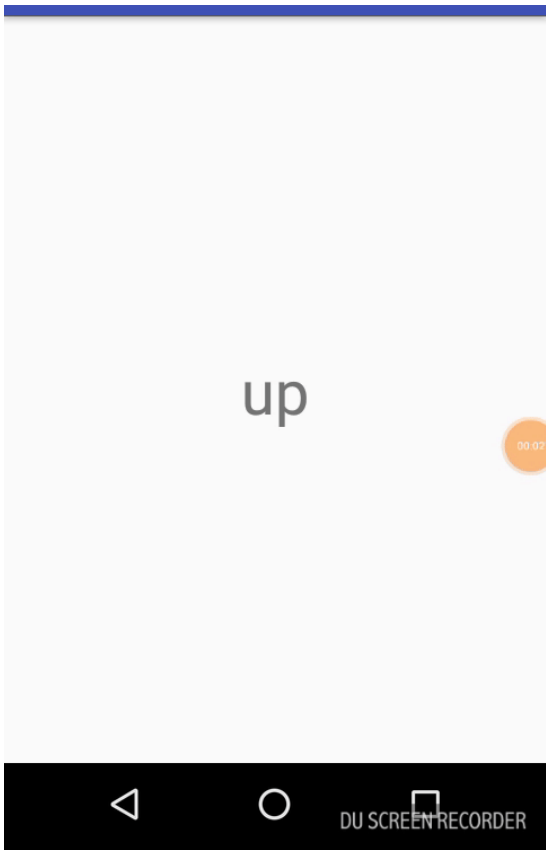
Speech Recognition Using TensorFlow

brijeshthumar / September 10, 2017 / TensorFlow

This tutorial will show you how to runs a simple speech recognition model built by the audio training tutorial. Listens for a small set of words, and display them in the UI when they are recognized.

It's important to know that real speech and audio recognition systems are much more complex, but like MNIST for images, it should give you a basic understanding of the techniques involved. Once you've completed this tutorial, you'll have a application that tries to classify a one second audio clip as either silence, an unknown word, "yes", "no", "up", "down", "left", "right", "on", "off", "stop", or "go".





1.Preparation

You can train your model on the desktop or on the laptop or on the server and then you can use that pre-trained model on our mobile device.So there's no training that would happen on the device the training would happen on our bigger machine either a server or our laptop.You can download a pretrained model from tensorflow.org

2. Adding Dependencies

The TensorFlow Inference Interface is available as a JCenter package and can be included quite simply in your android project with a couple of lines

in the project's build.gradle file:

```
allprojects {
    repositories {
        jcenter()
    }
}
```

Add the following dependency in app's build.gradle

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

This will tell Gradle to use the latest version of the TensorFlow AAR that has been released to https://bintray.com/google/tensorflow/tensorflow-android. You may replace the + with an explicit version label if you wish to use a specific release of TensorFlow in your app.

3.Add Pre-trained Model to Project

You need the **pre-trained** model and **label** file.You can download the model from here.Unzip this zip file, You will get `conv_actions_labels.txt` (label for objects) and `conv_actions_frozen.pb` (pre-trained model).

Put `conv_actions_labels.txt` and `conv_actions_frozen.pb` into `android/assets` directory.

4.Microphone Permission

To request microphone, you should be requesting `RECORD_AUDIO`

permission in your manifest file as below:

```
<uses-permission android:name="android.permission.RECORD_AUD
```

Since Android 6.0 Marshmallow, the application will not be granted any permission at installation time. Instead, the application has to ask the user for a permission one-by-one at runtime.

```
private void requestMicrophonePermission() {
        ActivityCompat.requestPermissions(MainActivity.this
                new String[]{android.Manifest.permission.RE
    }
@Override
public void onRequestPermissionsResult(int requestCode, Str
      if (requestCode == REQUEST_RECORD_AUDIO&& grantResult
                && grantResults[0] == PackageManager.PERMIS
            startRecording();
            startRecognition();
      }
 }
```

5.Recording Audio

The AudioRecord class manages the audio resources for Java applications to record audio from the audio input hardware of the platform. This is achieved by “pulling” (reading) the data from the AudioRecord object. The application is responsible for polling the AudioRecord object in time using read(short[], int, int).

```
 private void record() {
        android.os.Process.setThreadPriority(android.os.Pro

        // Estimate the buffer size we'll need for this dev
        int bufferSize =
                AudioRecord.getMinBufferSize(
                        SAMPLE_RATE, AudioFormat.CHANNEL_IN
```

```
        if (bufferSize == AudioRecord.ERROR || bufferSize =
            bufferSize = SAMPLE_RATE * 2;
        }
        short[] audioBuffer = new short[bufferSize / 2];

        AudioRecord record =
                new AudioRecord(
                        MediaRecorder.AudioSource.DEFAULT,
                        SAMPLE_RATE,
                        AudioFormat.CHANNEL_IN_MONO,
                        AudioFormat.ENCODING_PCM_16BIT,
                        bufferSize);

        if (record.getState() != AudioRecord.STATE_INITIALI
            Log.e(LOG_TAG, "Audio Record can't initialize!"
            return;
        }

        record.startRecording();

        Log.v(LOG_TAG, "Start recording");

        // Loop, gathering audio data and copying it to a r
        while (shouldContinue) {
            int numberRead = record.read(audioBuffer, 0, au
            int maxLength = recordingBuffer.length;
            int newRecordingOffset = recordingOffset + numb
            int secondCopyLength = Math.max(0, newRecording
            int firstCopyLength = numberRead - secondCopyLe
            // We store off all the data for the recognitio
            // thread will copy out of this buffer into its
            // lock, so this should be thread safe.
            recordingBufferLock.lock();
            try {
                System.arraycopy(audioBuffer, 0, recordingB
                System.arraycopy(audioBuffer, firstCopyLeng
                recordingOffset = newRecordingOffset % maxL
            } finally {
                recordingBufferLock.unlock();
            }
        }

        record.stop();
        record.release();
    }
```

6.Run TensorFlow Model

A `TensorFlowInferenceInterface` class that provides a smaller API surface suitable for inference and summarizing the performance of model execution.

```
private void recognize() {
    Log.v(LOG_TAG, "Start recognition");

    short[] inputBuffer = new short[RECORDING_LENGTH];
    float[] floatInputBuffer = new float[RECORDING_LENGTH];
    float[] outputScores = new float[labels.size()];
    String[] outputScoresNames = new String[]{OUTPUT_SCORES
    int[] sampleRateList = new int[]{SAMPLE_RATE};

    // Loop, grabbing recorded data and running the recogni
    while (shouldContinueRecognition) {
            // The recording thread places data in this rou
            // make sure there's no writing happening and t
            // local version.
       recordingBufferLock.lock();
       try {
           int maxLength = recordingBuffer.length;
           int firstCopyLength = maxLength - recordingOffse
           int secondCopyLength = recordingOffset;
           System.arraycopy(recordingBuffer, recordingOffse
           System.arraycopy(recordingBuffer, 0, inputBuffer
        } finally {
           recordingBufferLock.unlock();
        }

           // We need to feed in float values between -1.0
           // signed 16-bit inputs.
        for (int i = 0; i < RECORDING_LENGTH; ++i) {
            floatInputBuffer[i] = inputBuffer[i] / 32767.0
        }

           // Run the model.
       inferenceInterface.feed(SAMPLE_RATE_NAME, sampleRate
       inferenceInterface.feed(INPUT_DATA_NAME, floatInputB
       inferenceInterface.run(outputScoresNames);
       inferenceInterface.fetch(OUTPUT_SCORES_NAME, outputS

```

```
        // Use the smoother to figure out if we've had a rea
        long currentTime = System.currentTimeMillis();
        final RecognizeCommands.RecognitionResult result = r

        runOnUiThread(
                new Runnable() {
                    @Override
                    public void run() {
                      // If we do have a new command, highligh
                        if (!result.foundCommand.startsWith("
                            int labelIndex = -1;
                            for (int i = 0; i < labels.size
                             if (labels.get(i).equals(resul
                                    labelIndex = i;
                              }
                          }
                          label.setText(result.foundCommand)
                      }
                  }
                });
         try {
             // We don't need to run too frequently, so snoo
             Thread.sleep(MINIMUM_TIME_BETWEEN_SAMPLES_MS);
         } catch (InterruptedException e) {
          // Ignore
         }
     }

     Log.v(LOG_TAG, "End recognition");
}
```

7.Recognize Commands

RecognizeCommands class is fed the output of running the TensorFlow model over time, it averages the signals and returns information about a label when it has enough evidence to think that a recognized word has been found. The implementation is fairly small, just keeping track of the last few predictions and averaging them.

```
public RecognitionResult processLatestResults(float[] curre
        if (currentResults.length != labelsCount) {
```

```
            throw new RuntimeException(
                    "The results for recognition should con
                            + labelsCount
                            + " elements, but there are "
                            + currentResults.length);
        }

        if ((!previousResults.isEmpty()) && (currentTimeMS
            throw new RuntimeException(
                    "You must feed results in increasing ti
                            + currentTimeMS
                            + " that was earlier than the p
                            + previousResults.getFirst().fi
        }

        final int howManyResults = previousResults.size();
        // Ignore any results that are coming in too freque
        if (howManyResults > 1) {
            final long timeSinceMostRecent = currentTimeMS
            if (timeSinceMostRecent < minimumTimeBetweenSam
                return new RecognitionResult(previousTopLab
            }
        }

        // Add the latest results to the head of the queue.
        previousResults.addLast(new Pair<Long, float[]>(cur
        Log.d(TAG, currentResults + " " + currentTimeMS);

        // Prune any earlier results that are too old for t
        final long timeLimit = currentTimeMS - averageWindo
        while (previousResults.getFirst().first < timeLimit
            previousResults.removeFirst();
        }
        // If there are too few results, assume the result
        // bail.
        final long earliestTime = previousResults.getFirst(
        final long samplesDuration = currentTimeMS - earlie
        if ((howManyResults < minimumCount)
                || (samplesDuration < (averageWindowDuratio
            Log.v("RecognizeResult", "Too few results");
            return new RecognitionResult(previousTopLabel,
        }

        // Calculate the average score across all the resul
        float[] averageScores = new float[labelsCount];
        for (Pair<Long, float[]> previousResult : previousR
            final float[] scoresTensor = previousResult.sec
            int i = 0;
            while (i < scoresTensor.length) {
```

```
52                  averageScores[i] += scoresTensor[i] / howMa
53                  ++i;
54              }
55          }
56
57          // Sort the averaged results in descending score or
58          ScoreForSorting[] sortedAverageScores = new ScoreFo
59          for (int i = 0; i < labelsCount; ++i) {
60              sortedAverageScores[i] = new ScoreForSorting(av
61          }
62          Arrays.sort(sortedAverageScores);
63
64          // See if the latest top score is enough to trigger
65          final int currentTopIndex = sortedAverageScores[0].
66          final String currentTopLabel = labels.get(currentTo
67          final float currentTopScore = sortedAverageScores[0
68          // If we've recently had another label trigger, ass
69          // soon afterwards is a bad result.
70          long timeSinceLastTop;
71          if (previousTopLabel.equals(SILENCE_LABEL) || (prev
72              timeSinceLastTop = Long.MAX_VALUE;
73          } else {
74              timeSinceLastTop = currentTimeMS - previousTopL
75          }
76          boolean isNewCommand;
77          if ((currentTopScore > detectionThreshold) && (time
78              previousTopLabel = currentTopLabel;
79              previousTopLabelTime = currentTimeMS;
80              previousTopLabelScore = currentTopScore;
81              isNewCommand = true;
82          } else {
83              isNewCommand = false;
84          }
85          return new RecognitionResult(currentTopLabel, curre
86      }
```

The demo app updates its UI of results automatically based on the labels text file you copy into assets alongside your frozen graph, which means you can easily try out different models without needing to make any code changes. You will need to updaye `LABEL_FILENAME` and `MODEL_FILENAME` to point to the files you've added if you change the paths though.

8.conclusion

You can easily replace it with a model you've trained yourself. If you do this, you'll need to make sure that the constants in the main MainActivity Java source file like `SAMPLE_RATE` and `SAMPLE_DURATION` match any changes you've made to the defaults while training. You'll also see that there's a Java version of the RecognizeCommands module that's very similar to the C++ version in this tutorial. If you've tweaked parameters for that, you can also update them in MainActivity to get the same results as in your server testing.

Download this project from GitHub

Related Post

Android TensorFlow Machine Learning

Google Cloud Speech API in Android APP

Share this post: on Twitter on Facebook on Google+

Previous Article
‹ EmojiCompat Support Library

Next Article
How to Create Instant app from Existing App ›

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

Recent Posts

- How Location API Works in android
- Firebase Crashlytics
- How to use DateTime datatype in SQLite Using Room
- Cloud Firestore Database for Web Application
- ConstraintLayout 1.1.0: Circular Positioning
- Firestore Security With Firebase Authentication
- Firestore Document Database Data Model
- Firebase Firestore Database for Android Application
- Rest API Pagination with Paging Library.
- Architecture Components:Paging Library
- Autosizing TextViews Using Support Library 26.0
- How to Create Instant app from Existing App
- Speech Recognition Using TensorFlow
- EmojiCompat Support Library
- Fast Scrolling in RecyclerView Using Support Library 26

Categories

- architecture (7)
- Connectivity (2)
- Firebase (8)
- kotlin (1)
- Layout (1)
- Library (1)
- Location (2)
- Machine Learning APIs (3)
- Performance (2)
- TensorFlow (6)
- Uncategorized (13)

Archives

- December 2017
- November 2017
- October 2017
- September 2017
- August 2017
- July 2017
- June 2017
- May 2017
- April 2017

Proudly powered by androidkt