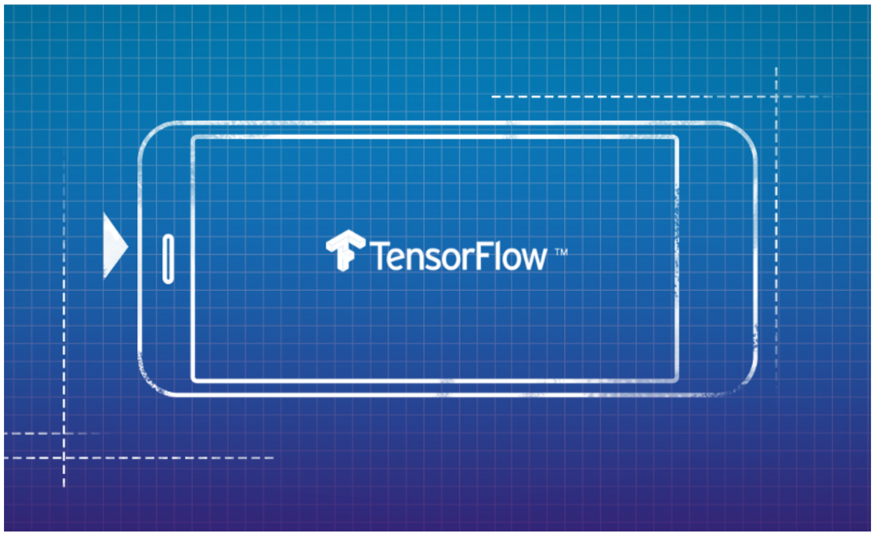
Images haven't loaded yet. Please exit printing, wait for images to load, and try to print again.

Tech Lead @ JoyTunes. iOS, Android, Deep Learning. Software Craftsman. Hands on dad.
Mar 29, 2017 · 6 min read

Deploying a TensorFlow model to Android

a guide written in blood 💉



There are a lot of tutorials and material out there about TensorFlow, but information about deploying a TensorFlow model into a mobile app (without a server-side component) is very lacking.

At JoyTunes, as part of our constant work on improving the amazing MusicSense™ piano recognition engine, I wanted to create a POC for deploying a model created in TensorFlow in our research environment directly to a production Android app. Since I spent quite a lot of time figuring it out myself, running into a lot of pitfalls, I decided to dump my experience into a post others like me might find useful.

· · ·

Existing guides

First, you will want to look at the Android example on github. It's the only official example by TensorFlow explaining how to run a model on Android, and it's a good place to learn from.

Second, there are two very relevant posts by Amit Shekhar, which unfortunately I stumbled upon only a week after starting this onerous journey.

The first is another example of an Android app running a TF model (the same one from the official TensorFlow example), but with a step-by-step explanation of how it was built (exactly what my post talks about but with a very sunny-side scenario of things):

Android TensorFlow Machine Learning Example

Machine Learning: Integrating Tensorflow in Android

blog.mindorks.com

The second is an explanation of how to create a custom model that can be deployed in the same manner:

Creating Custom Model For Android Using TensorFlow

As I had promised in my previous article on building TensorFlow for Android that I will be writing an article on How to...

blog.mindorks.com

It may very well be possible that these two guides alone must suffice for your needs. But in case you are running into issues, I come to the rescue with some pitfalls & troubleshooting.

. . .

My step-by-step guide

1. Obtain a frozen version of your model as protobuf (.pb)

I'm not going to go over this part too much, since this part was already done for me in the research environment. Two good posts about how to do it is second post by Amit above, or the following one:

TensorFlow: How to freeze a model and serve it with a python API

We are going to explore two parts of using a ML model in production:

blog.metaflow.fr

Place the created model.pb file in your app's assets directory.

2. Double check the frozen model nodes

First pitfall: your model freezing code might not work exactly as expected.

To use the model in code (described in step 6), you'll want to know the exact node names and shape for input and output.

Even if you're sure about what they are, maybe tensorflow had to rename a node (e.g. add _1 to the name because of duplications in the graph) or there was a mistake in defining something.

To double check yourself, load the graph in python and have a look:

```
>>> import tensorflow as tf

>>> g = tf.GraphDef()

>>> g.ParseFromString(open("path/to/mymodel.pb",
"rb").read())

>>> [n for n in g.node if n.name.find("input") != -1] # same
for output or any other node you want to make sure is ok
```

Make sure node description is what you expect it to be and take note of the input shape and the exact input/output node names. You'll need them in your app's code later.

3. Check that you support all necessary OpKernels

EDIT June '17: Since writing this section Pow was added as an Op by default, and also, maybe if you apply Selective Registration (as explained in my post) you can skip this part altogether, didn't check

This step might be redundant for you, so maybe you want to come back to it later only if you run into problems.

For us, we were using the operation "Pow" in our model, and it didn't work. The solution described here is based on the following StackOverflow thread:

No OpKernel was registered to support Op 'Pow' on Android

I use custom inception model of Android and when I try session->Run() I receive following error: No OpKernel was...

stackoverflow.com

There might be other tensorflow kernels that your model is using and aren't included by default in the android library (for us it ended up being only Pow but I wanted to make sure). To make sure, you can run in your python environment (after loading the graph from model file as described above):

```
>>> ops = set([n.op for n in g.node])
```

Now run `print “ “.join(ops)` to get a command-line friendly list of operators. Then you can see in which source file each operator is declared by running (in bash):

```
cd <tensorflow_repo>/tensorflow/core/kernels
for i in <op_list>
do
    echo ====
    echo $i
    echo ====
    grep -Rl \”$i\” .
done
```

You'll sometimes get more than one file in the answer, this usually means it is a common operator and you don't need to worry about but in any case you should be able to tell which of the files is probably the one that is the primary one defining the given operator.

Next step is to edit `tensorflow/core/kernels/BUILD` and to check that the relevant file declaring the op is present under one of the `android_XXX_ops` filegroups, If it isn't, add it to one (e.g. `android_extended_ops_group1`)

4. Build necessary .so and .jar libraries

Build instructions are described well in the first post by Amit I linked to above.

EDIT June ‘17: Since writing this, I discovered another important step you might strongly want to consider — applying Selective Registration to the *built .so to significantly slim down it's size. Read about it in the following post:*

Slimming down TensorFlow's impact on your .APK size

By applying selective registration

medium.com

In any case, I took the liberty of creating a nice script that executes the whole build for all relevant architectures without the need to edit the WORKSPACE file. The script can be found in a JoyTunes fork of tensorflow, forked from latest stable release, v1.1.0 as of writing this.

The specific script can be found here (assumes the WORKSPACE file in configured as in this fork):

prepare_android_deployment.sh

JoyTunes fork of tensorflow

github.com

If all is well, the script should do everything for you and create a `deploy_android` dir ready with a jar and native libraries compiled for the 4 common Android architectures. You should copy its content to your app's project.

In case you didn't change anything in step 3, you can also consider downloading the jar and the 4 relevant .so files from the tensorflow nightly build binaries instead. However, I don't recommend it for reasons mentioned in the troubleshooting section below.

5. Edit your app's gradle file:

After copying the `tensorflow` directory with built libraries from the previous step, you should edit your gradle file to make sure it finds and loads these libraries. The difference should look something like this:

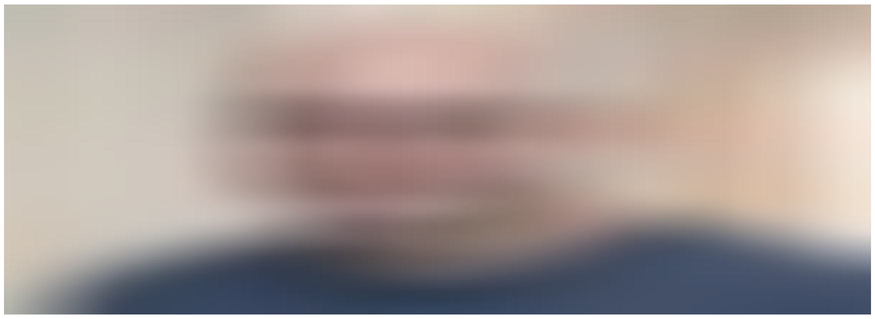
```
repositories {
    flatDir {
        dirs "tensorflow"
    }
}

dependencies {
    compile(name:'libandroid_tensorflow_inference_java',
ext:'jar')
}

android {
    sourceSets {
        jniLibs.srcDirs = ['libs',
'tensorflow/prebuiltLibs']
    }
}
```

6. Load and run the model in your android's app code

Now that the tensorflow libraries are ready to be invoked from your app, you can use them as follows:



7. Tell proguard not to obfuscate TensorFlow

This is something I discovered only when running in release after publishing this post originally.

You'll want to edit your `proguard-project.txt` to make sure TensorFlow JNI calls will work well:

```
-keep class org.tensorflow.** { *; }
```

That's it! Now run your app and hope for the best :)

. . .

Troubleshooting

- Make sure you have the 12b version of the NDK. Trying to build tensorflow with the latest NDK version didn't work for me.
- The tensorflow version you used for creating the model must be the same version you built the library with. This is one of the reasons using prebuilt binaries from the nightly build didn't do the trick for me.
- Look for "tensorflow" in logcat to find more errors loading the model or running it. I already mentioned ways to avoid the ones that occurred to me, such as wrong names for input/output nodes, wrong shape or data type (e.g. double array instead of float array), and missing support for an operation.
- Bumped into anything else? Ping me here or on twitter and I'll try to help

. . .

I hope you found this post useful. Good luck!