This is an example of an Android app that uses OpenCV DNN module to load a Caffe model and predict an image. It implements the tutorial from http://docs.opencv.org/master/d5/de7/tutorial_dnn_googlenet.html#gsc.tab=0

**16** commits | **2** branches | **0** releases | **1** contributor

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find file** | **Clone or download**

**alexkarargyris** Updated code to allow app to import an image from camera or photo gal… ··· Latest commit `b6b96f4` on 25 Jan 2016

| | | |
|---|---|---|
| .idea | Updated code to allow app to import an image from camera or photo gal… | 2 years ago |
| app | Updated code to allow app to import an image from camera or photo gal… | 2 years ago |
| gradle/wrapper | Initial commit | 2 years ago |
| opencv_java | Updated code to allow app to import an image from camera or photo gal… | 2 years ago |
| .gitignore | Initial commit | 2 years ago |
| README.md | Updated Readme | 2 years ago |
| build.gradle | Initial commit | 2 years ago |
| gradle.properties | Initial commit | 2 years ago |
| gradlew | Initial commit | 2 years ago |
| gradlew.bat | Initial commit | 2 years ago |
| settings.gradle | Initial commit | 2 years ago |

**README.md**

# Caffe OpenCV Android App

This is an example of an Android app that uses OpenCV DNN module to load a Caffe model and predict an image. It is basically the OpenCV tutorial for DNN: http://docs.opencv.org/master/d5/de7/tutorial_dnn_googlenet.html#gsc.tab=0

NOTE: Jump to the end of this tutorial (https://github.com/alexkarargyris/Caffe_OpenCV_Android_App#c-alternative-fast-way-to-run-this-project) to kick off quickly!

##A. Prerequisites

## OpenCV

- Download OpenCV source code from here: https://github.com/Itseez/opencv
- Download OpenCV external modules from here: https://github.com/Itseez/opencv_contrib

## Android Studio

- Download & install Android Studio: http://developer.android.com/sdk/index.html

This will install Android SDK.

##Requirements

###1. Install NDK Bundle

- Open Android Studio and navigate Tools->Android->SDK Manager. Choose "SDK Tools" then tick "Android NDK". Finally "Apply" and hit "OK" to close. You may need to restart Android Studio. Alt text

###2. Install Android NDK

**IMPORTANT! MAKE SURE THAT YOUR Android NDK IS ON YOUR $PATH SO THAT YOU CAN USE IT FROM ANYWHERE (e.g. bash_profile on Mac: https://github.com/9miao/CrossApp/wiki/Android-Development-Environment-Configuration-in-Mac-OS-X#2bash_profile-file-configuration)**

###3. Build Android OpenCV SDK with extra modules for Android Unfortunately the Deep Neural Network (DNN) module for OpenCV is not part of the main OpenCV distribution yet.

It resides in the extra modules. So we have to build OpenCV along with extra modules. For this we need to build them together.

Here is how:

- Open your file explorer (e.g. Finder on Mac) and navigate to `/Users/alexandroskarargyris/Downloads/opencv/platforms/scripts` . Replace with your own path.
- Open `cmake_android_arm.sh` with a text editor (i.e. Textedit on Mac)
- Edit it to look like this:

```
#!/bin/sh
cd `dirname $0`/..

mkdir -p build_android_arm
cd build_android_arm

cmake -DOPENCV_EXTRA_MODULES_PATH=/Users/alexandroskarargyris/Downloads/opencv_contrib/modules -DCMAKE_BUIL
```

where `-DOPENCV_EXTRA_MODULES_PATH=` points to where the OpenCV external modules reside. So replace it with your own path. Save and exit your text editor.

- Open terminal:

- After making is finished then run:

```
$ cd build_android_arm
$ make install
```

This process will install the Android OpenCV SDK under `/Users/alexandroskarargyris/Downloads/opencv/platforms/build_android_arm/install` folder. That concludes the most important part: building the Android OpenCV SDK with extra modules (e.g. DNN) from source code.

##B. App Development

For this part I used the wonderful instructions from https://github.com/quanhua92/NDK_OpenCV_AndroidStudio

###1. Create OpenCV Gradle Project

- Open Android Studio
- Import a Project
- Navigate to your OpenCV Android SDK location (see A.3 step above). My SDK's location is at `/Users/alexandroskarargyris/Downloads/opencv/platforms/build_android_arm/install`. Find `SDK/java` and import it


- Select the name (e.g. OpenCV-Java)where you want to save the new project and don't forget to build it
- That's it. Close Project.

- Open Android Studio
- Create a new project (i.e MyApplication)
- Let's import OpenCV and its libraries: Go `File->New->Import Module`. Click `...` in the new window and navigate to the location of your OpenCV Java project from step B.1 above.


Then type in module name: `opencv_java`:


This will import OpenCV Java to our project. Finally go to `opencv-java->build.gradle` to configure the file to look like below:


- Add the OpenCV libraries by draggin & dropping the `libs` folder (i.e. `/Users/alexandroskarargyris/Downloads/opencv/platforms/build_android_arm/install/sdk/native/`) to your project. Don't forget to rename to `jniLibs`. See figure below:


- In the project structure navigate to app->src->main->java->myapplication and add a new class (e.g. NativeClass) with the following method:

```
public class NativeClass {
   public native static String getStringFromNative();
```


- Open a terminal, change to app/src/main and run the following command:

```
javah -d jni -classpath ../../build/intermediates/classes/debug/
com.example.alexandroskarargyris.myapplication.NativeClass
```

This command basically creates the header file for native C/C++ to allow access to C/C++ OpenCV calls.

- Back to Android Studio you will find a new folder `jni` as well as the the header: `com_example_alexandroskarargyris_myapplication_NativeClass.h` . Open it and you will see the following method:

```
JNIEXPORT jstring JNICALL Java_com_example_alexandroskarargyris_myapplication_NativeClass_getStringFromNati
        (JNIEnv * env, jobject obj);
```


This is the C/C++ method that `getStringFromNative()` in `NativeClass` is going to call. So let's create a .cpp to have our C++ code for this method. For simplicity please go ahead and copy the code from the repository: https://github.com/alexkarargyris/Caffe_OpenCV_Android_App/blob/master/app/src/main/jni/com_example_alexandroskarargyris_myapplication_NativeClass.cpp This is the main code for runnning the DNN using OpenCV. You will notice that I have left many C++ calls (e.g. `std::cerr <<` ) from the original tutorial code. The reason is to show how easy it is to migrate it to your Android project.

- Add `Android.mk` file under `jni` folder and modify it to look like the figure below:

- Add `Application.mk` file under `jni` folder and modify it to look like the figure below:
- Modify the app's `build.gradle` to look like this:


This tells the app where to look for NDK, the JNI files, the configuration files (i.e. Android.mk) etc. It is very important.

- Now let's go to `MainActivity.java` tell it load the OpenCV library and our JNI library (e.g. MyLib) by adding the following:

```
static {
    System.loadLibrary("MyLib");
    System.loadLibrary("opencv_java3");
}
```

Also in the figure below you can see how I call the `tv.setText(NativeClass.getStringFromNative());` which in return runs `NativeClass` which in return runs `Java_com_example_alexandroskarargyris_myapplication_NativeClass_getStringFromNative` from `com_example_alexandroskarargyris_myapplication_NativeClass.cpp`

This will print the classifier's predicted class name in a text view I setup in `activity_main.xml` . But I suppose you know how to do this.

- Finally you need to load up the Caffe models to your device (virtual or real). You can download them from the tutorial's page: http://docs.opencv.org/master/d5/de7/tutorial_dnn_googlenet.html#gsc.tab=0 . I placed an image of a space shuttle in the app's resources to let the classifier predict it.

If you reach so far and played with the above code then you learned quite a lot things. Regardless if your efforts were unsuccessful. The way to cheat this tutorial is to import it directly form Github to your Android Studio. To run it you need to have:

1. Build OpenCV with extra modules as described in Requirements
2. Install NDK as described in Requirements
3. Modify `Android.mk` , `Application.mk` , `build.gradle` with your own file paths (see B.2)