Tensorflow在ANROID系统实际应用操作说明

陈国强10034816

TensorFlow是一个开源的用于机器学习的库，是由谷歌开发的并用在他们的多个项目中。

本文打算从一个Android 开发者的角度切入，看看构建一个基于 TensorFlow 的 Android 应用的完整流程,并在目前现有不同机型上进行实际运行使用。

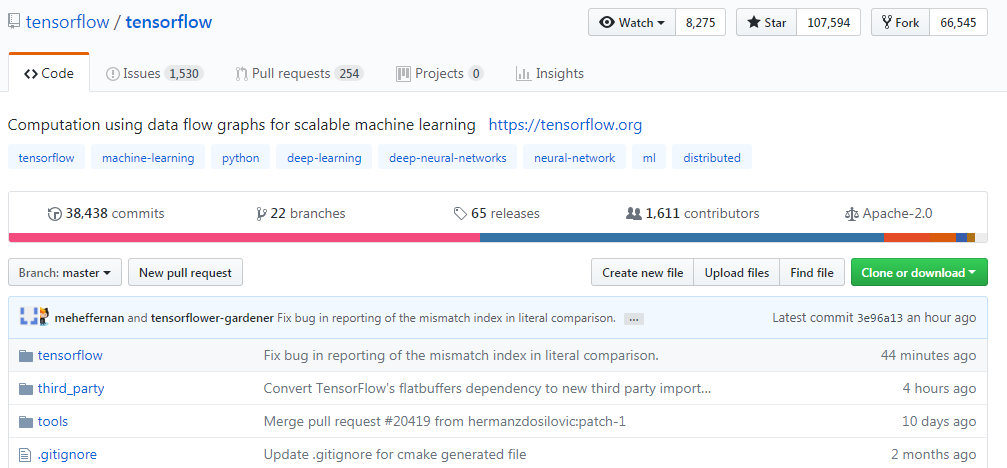
# 0 准备工作

## 0.1 GITHUB账号，CHROME浏览器

安装CHROME浏览器，进入GITHUB网站，这是一个重要网站，申请相应账号，创建自己的代码库。GITHUB账号申请后即可以下载代码，或者CLONE相应代码库到自己账号，下面为相应TENSORFLOW的GITHUB代码库地址。

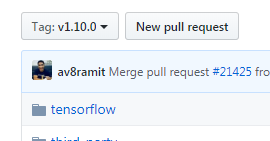


进入TENSORFLOW代码库后：



这里BRANCH类似于分支概念，可以根据需要获取相应稳定版本代码。

本文在这里，选择最新TAGS：1.10.0获取相应DEMO代码



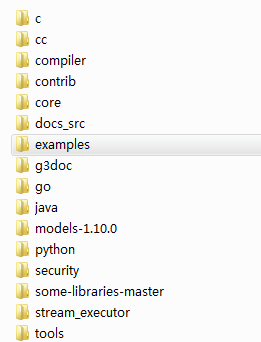
CHROME浏览器这个为常用浏览器，不做进一步介绍。

## 0.2 ANDROID STUDIO安装

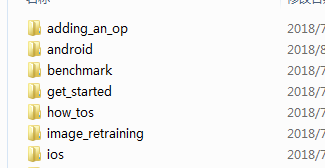
STUDIO安装及SDK,NDK下载及配置，由于是常用工具，不做进一步介绍。

# 1 工程导入

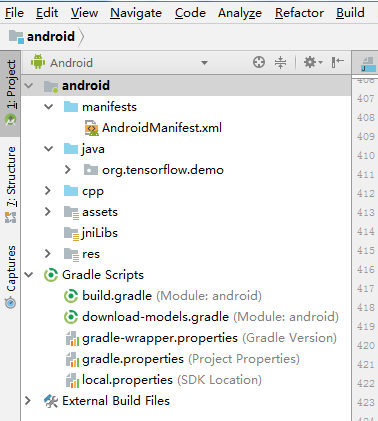
**源码完成下载：**



源码目录结构如图，相应ADNROID部分从examples里进入。



这里android目录即是我们所需要用到的ADNROID工程目录，使用Android Studio导入该工程即可。

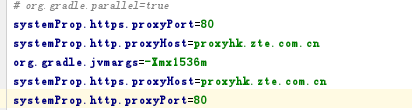


# 2 关键配置

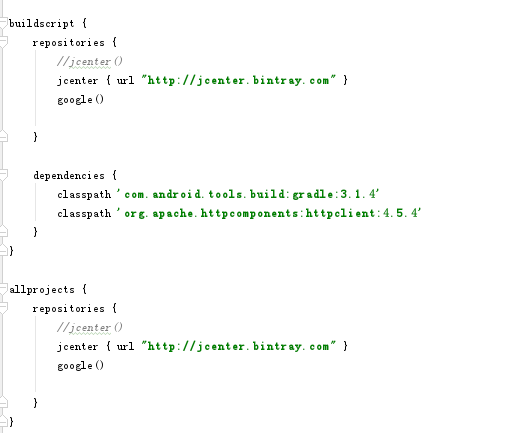
由于我们处于公司内网环境，所以有些配置需要特别关注处理，否则工程无法正常编译运行。

2.1，添加gradle.properties属性文件

这里主要是公司内网PROXY代码配置



2.2，build.gradle文件根据内网环境改写



2.3，同样download-models.gradle类似依赖库位置需要相应改写



2.4，build.gradle文件配置本地库依赖



这里，需要说明，前期为了DEMO代码运行，了解运行流程，我们可以将NATIVE依赖库不编译进去所以配置为NONE。由于配置为NONE，某些功能比如目标识别会有功能影响。

# 3 DEMO实际运行

3.1，完成DEMO代码编译

根据如上相应配置，一般情况下Android Studio可以较为顺利完成代码编译，并输出演示APK。当然，个别机器系统等情况需要具体分析。

3.2，APK运行

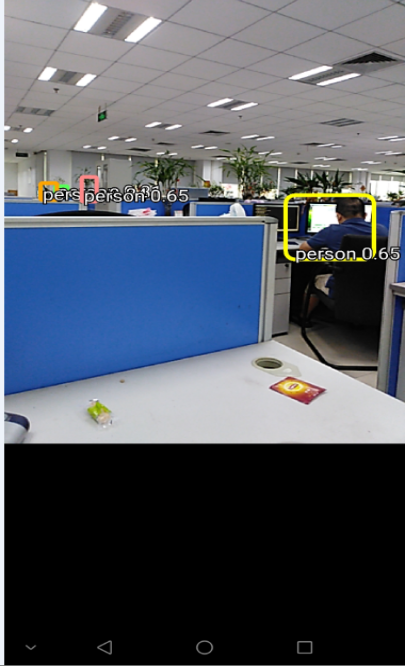
APK主要演示了4个功能

3.2.1，分类识别：实时分析相机图像数据帧来识别物品类别

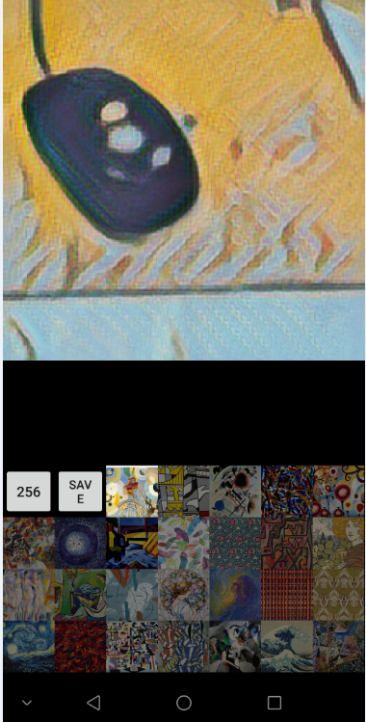


由于是实时获取当前摄像头预览界面图像数据帧，所以随着手的晃动，数据结果都会实时变化。

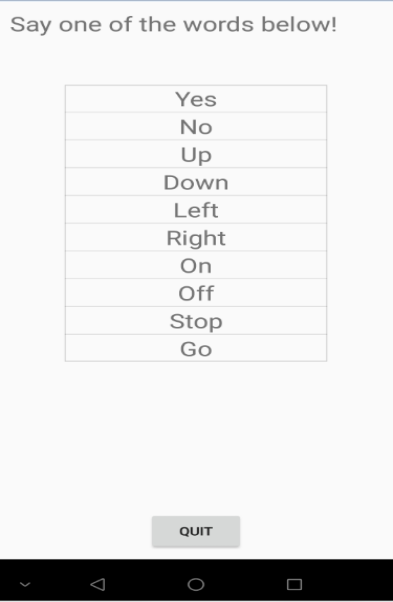
3.2.2，目标识别：实时分析相机图像数据帧来**实时目标检测**



**3.2.3，**图片风格转化:实时分析相机图像数据帧**进行图像风格迁移**



**3.2.4,语音识别：实时识别输入语音数据识别单词**

实时识别用户说出的单词，并在表格中高亮显示

# 4对于分类识别的模型替换实践

**前面部分已经对DEMO应用的集成，编译，输出，使用进行了基本流程介绍。目前编译输出的APK在P639S10(MT6739),P840V71(MSN8940),P450S10(SDM450)这几种机型上进行实际使用均可正常运行。其中MT6739平台芯片及摄像能力相对较差，实时图像识别略有迟滞，其余平台均可流畅使用。**

**下面重点对分类识别的TENSORFLOW使用模型部分进行相应实践介绍。**

****简单归纳来说，机器学习分为两个阶段：****

训练阶段：通过对有标签的样本数据进行训练得到合适的最佳模型

推测阶段：根据模型对无标签的样本数据进行推段得到结果（标签）

这里训练阶段即模型的训练获取。DEMO应用里面的分类识别，所使用模型为默认附带，那么我们往往需要根据实际应用的场景情况需要相应的场景模型。

下面就是一些实际模型的训练获取及模型导入使用过程。

4.1，准备工作

OS：WIN7，完成TENSORFLOW，PYTHON的安装

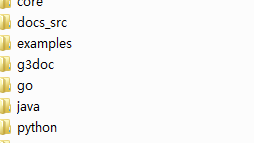
目前使用TENSORFLOW VERSION为1.9.0（最新的应该是1.10），PYTHON VERSION为3.5.2



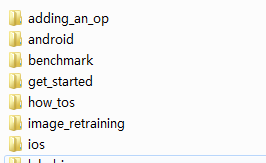
安装完PYTHON后，一般使用PIP INSTALL进行TESORFLOW安装

之后需要进行TENSORFLOW\_HUB安装。

4.2，实际进行模型训练



在examples目录下的image\_retraining\retrain.py文件就是关键的训练执行文件

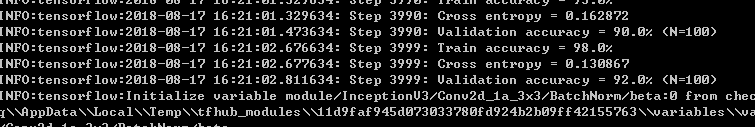


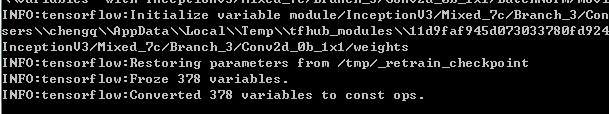
**我们在**image\_retraining同目录下新建flower\_photos目录并放入相应各种类花卉的原始数据图片。

通过阅读代码retrain.py，我们了解到基本的默认模型训练参数都有预设默认值，我们需要的做是输入图片数据路径即可以开始模型训练。

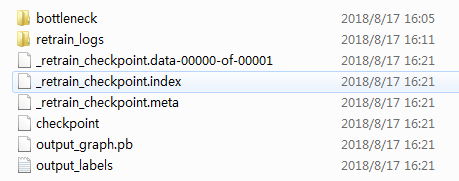
python retrain.py ^

--image\_dir F:/tensor0720/tensorflow190/tensorflow-1.9.0/tensorflow/examples/image\_retraining/tmp/flower\_photos

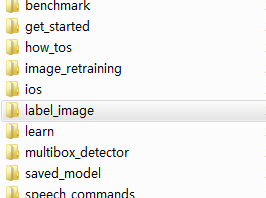




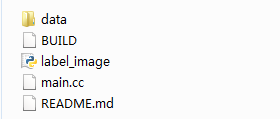
训练完成，则会生成tmp目录，目录里面的output\_graph.pb和output\_labels.txt即为相应输出的模型使用文件



4.3，对经过训练的模型进行验证测试

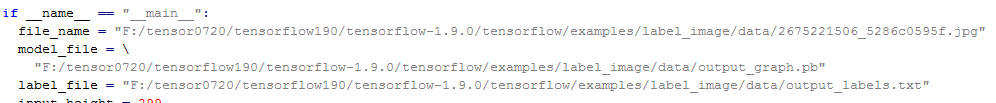


在examples目录下的label\_image目录里面，



关键的是label\_image.py文件。

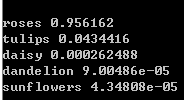
这里，在该文件的MAIN入口，是相应模型文件，测试图片文件的输入位置，这里我们进行模型和测试图片的导入

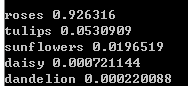


我们从网络随机获取两个ROSE图片，填入相应路径，运行



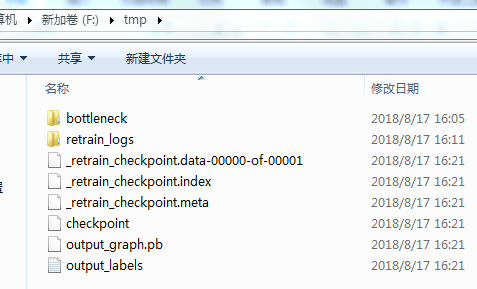
得到输出结果：说明ROSE的可信度还是比较符合预期，但是对于不同的图片由于背景颜色，图片中特征性等的不同，识别率还是有所差别。当然模型可以不断根据实际情况进行优化改进。而如果是DEMO默认附带而不是专门训练模型，则识别率相差非常大，这里不再描述。



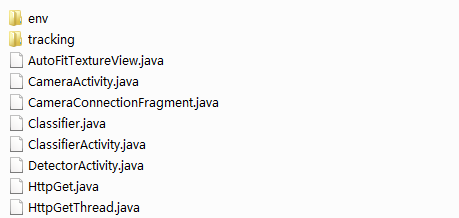


4.4，将训练过的花卉模型导入ANDROID工程进行实时场景测试

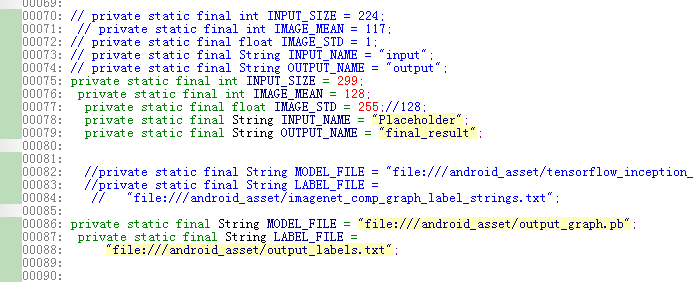
训练完成，会生产一个tmp文件夹，相应我们所需要模型文件即在该文件夹



然后进入代码目录的classifierActivity.java文件



根据代码说明，修改输入模型文件的相应路径并进行相应参数配置



这样即完成相应配置，之后重新进行工程BUILD，输出新的APK，并将APK导入手机。

打开电脑中的该图片，实时场景识别，发现识别率相比电脑中进行图片识别有较大差距。特别是镜头拉近拉远或者角度等不同都会产生影响。

也就是说实时场景识别，面临着实际环境中的环境因素干扰，如背景元素，光照远近强度，浏览角度等都会直接影响最终识别率结果，相比电脑图片识别需要进一步考虑优化的参数要复杂很多。还需要有逐步优化的过程。

或者说我们模型训练所使用的原始数据图片需要增加实际场景的数据图片以便完善模型。

# 5总结

# 通过上述段落，已经可以较为完整的完成整个演示流程，并且完成模型训练导入等后续开展。

基于此，在终端上面可以拓展非常丰富的基于AI的智能应用，这也是后续的发展方向。