Tflite

一 基本介绍

代码路径:

tensorflow-r1.11\tensorflow\contrib\lite\java\demo

应用效果:



**一些基本概念**

**何为nnapi**

TFLite 是 Google I/O 2017 推出的面向移动端和嵌入式的神经网络计算框架，于2017年11月5日发布开发者预览版本 (developer preview)。相比与 TensorFlow，它有着这样一些优势：

1. 轻量级。如上所述，通过 TFLite 生成的链接库体积很小；
2. 没有太多依赖。TensorFlow Mobile 的编译依赖于 protobuf 等库，而 tflite 则不需要大的依赖库；
3. 可以用上移动端硬件加速。TFLite 可以通过 Android Neural Networks API (NNAPI) 进行硬件加速，只要加速芯片支持 NNAPI，就能够为 TFLite 加速。不过目前在大多数 Android 手机上，Tflite 还是运行在 CPU 上的。

**关于quantization和float**

目前我们使用神经网络的时候，通常都是使用浮点数，这是保持模型准确性的最简单方法，并且GPU可以很好地加速这些计算，所以很自然的是对其他数字格式没有太多关注。

但是实际使用中，有很多模型被部署在商业应用中，模型的前向推导计算需求随着用户数量的增加而成正比，这意味着纯粹的前向推理效率已经成为很多团队的热门关注点。

这就是 Quantization 之所以出现的原因，它以比32位浮点数更紧凑的格式来存储数字，并对它们进行计算。本文重点放在8位固定点数值格式上，原因在后面我会详细讨论。

为什么要 Quantization 量化？

第一个原因就是减小模型占用空间，例如，神经网络模型可占用大量磁盘空间，原始AlexNet的浮点格式超过200 MB。几乎所有的大小都被神经连接的权重所占据，因为在单个模型中经常有数百万个权重参数，并且它们都是略有不同的浮点数，所以像zip这样简单的压缩格式并不能很好地压缩它们。并且它们有一个较好的特征，那就是在每一层内的权重倾向于在一定范围内正态分布，例如-3.0至6.0。

量化最简单的方式是通过先存储每个层的最小值和最大值，然后将每个浮点值压缩成一个8位整数，在最大值、最小值范围内空间线性划分 256 段，每段用一个唯一的 8-bit 整数表示在该段内的实数值； 例如-3.0到6.0的范围，0字节将代表-3.0，255代表6.0，128代表约1.5。简单的使用8位整数存储权重就意味着可以从磁盘上压缩大约 75％ 的文件大小，然后在加载后转换回浮点数，以便现有的浮点代码可以没有任何改变地工作。

另一个原因是如果不像上述方式只是使用8位整数来存储模型，而是完全用8位整数执行输入和输出，以及执行推理计算，可以大大减少计算资源；但这样做要困难得多，因为需要改变模型的所有代码，但是这样做有很多好处。获取8位值只需要浮点内存带宽的25％，因此可以更好地使用缓存并避免RAM访问出现瓶颈，也可以在每个时钟周期执行更多的SIMD操作。另外如果有一个可以加速8位计算的DSP芯片，将可以提供更多优势。

将计算结果移至8位可帮助您更快速地运行模型，并使用更少的功耗（这对于移动设备尤为重要），它也为许多无法高效运行浮点代码的嵌入式系统打开了大门，因此可以在物联网设备中启用大量的应用程序。

**为什么不直接用8位的权重进行较低的精度训练？**

一些实验结果表明，必须使用高于8比特的数据来处理反向传播和梯度的数值。并且我们已经有很多可以使用的浮点数模型，所以可以非常方便的直接转换它们。

参考: <https://zhuanlan.zhihu.com/p/33535898>

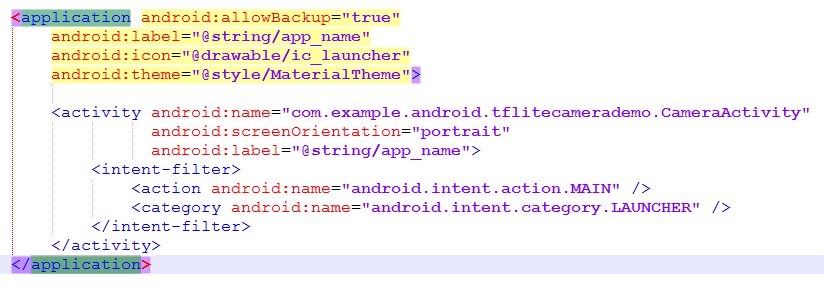
压缩模型的toco是什么

备忘: toco工具目前不支持windows,用在压缩tflite模型的时候.

二 分析

2.1 应用入口

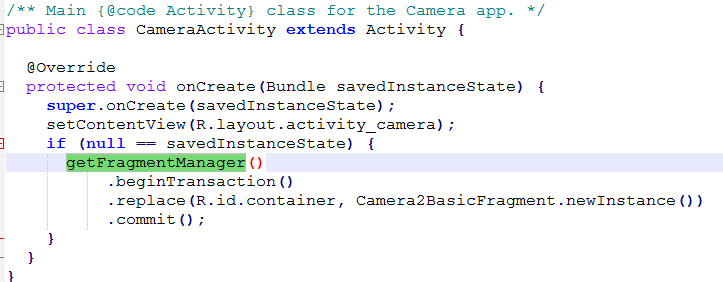
在androidManifest.xml中,关键词: **action.MAIN, category.LAUNCHER, CameraActivity**



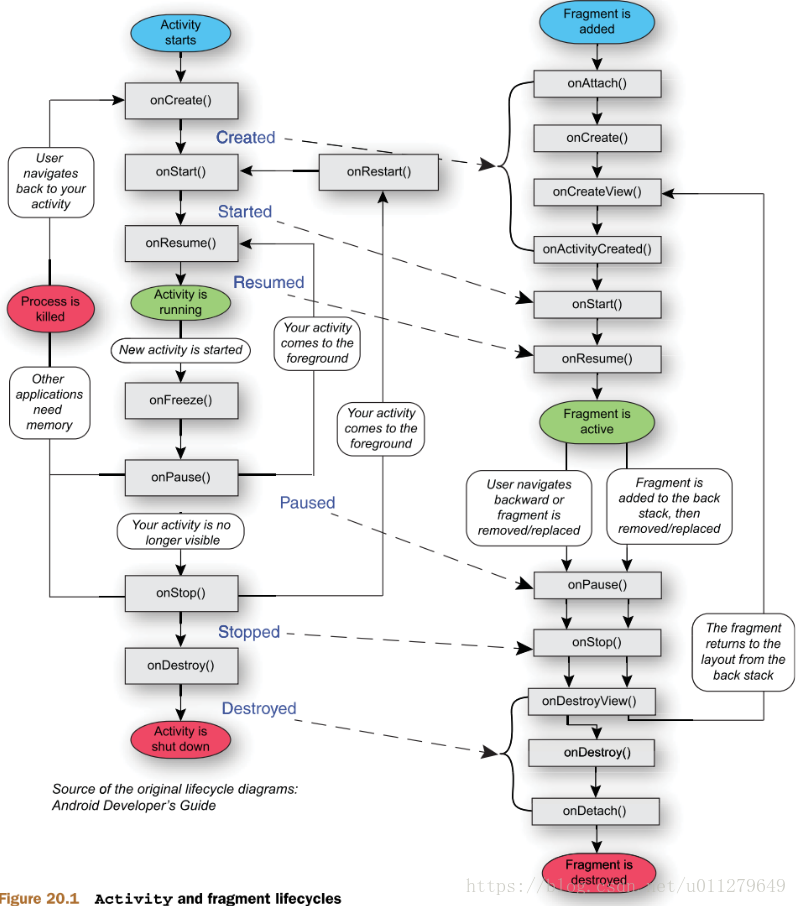
2.2 大体流程

在CameraActivity.java中.关键词**Camera2BasicFragment.newInstance()**.

要管理fragment们，需使用FragmentManager，要获取它，需在activity中调用方法getFragmentManager().



Fragment的生命周期,如下图:



Fragment的关键函数

1. onCreateView
2. onViewCreated
3. onActivityCreated
4. onResume, onPause, onDestroy

surfaceTextureListener关键函数,在onResume时候用到.

1. onSurfaceTextureAvailable

打开相机openCamera

1. onSurfaceTextureSizeChanged

配置尺寸变化configureTransform

CameraDevice.StateCallback stateCallback的关键函数:

1. onOpened

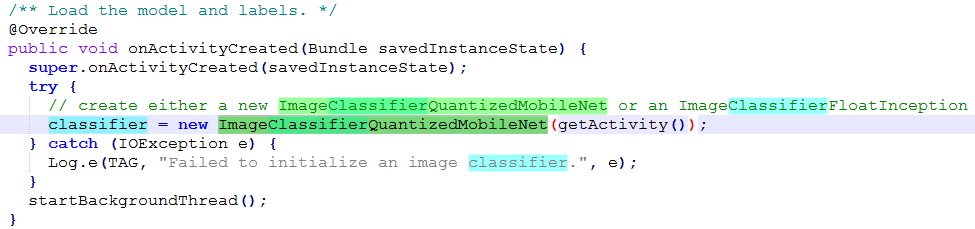
创建createCameraPreviewSession().

1. onDisconnected
2. onError

这两个是关闭的一些处理.

重点在onViewCreated. onActivityCreated.

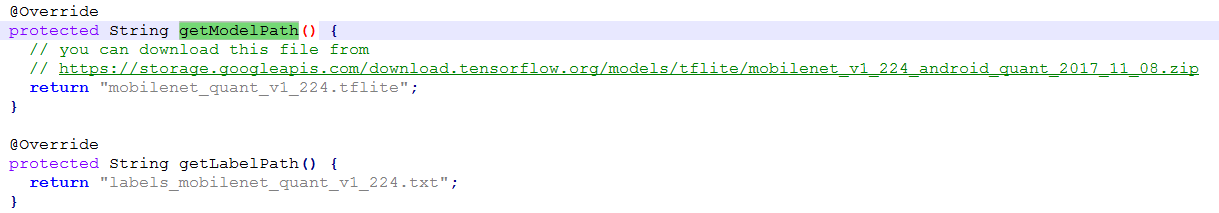
onActivityCreated创建图像分类器java对象.



ImageClassifierQuantizedMobileNet在ImageClassifierQuantizedMobileNet.java中实现.继承于ImageClassifier,并实现(重写)其getModelPath, getLabelPath函数.

描述着tflite模型名字和label名.

getImageSizeX, getImageSizeY描述了模型输入img的尺寸.



TextureView.SurfaceTextureListener

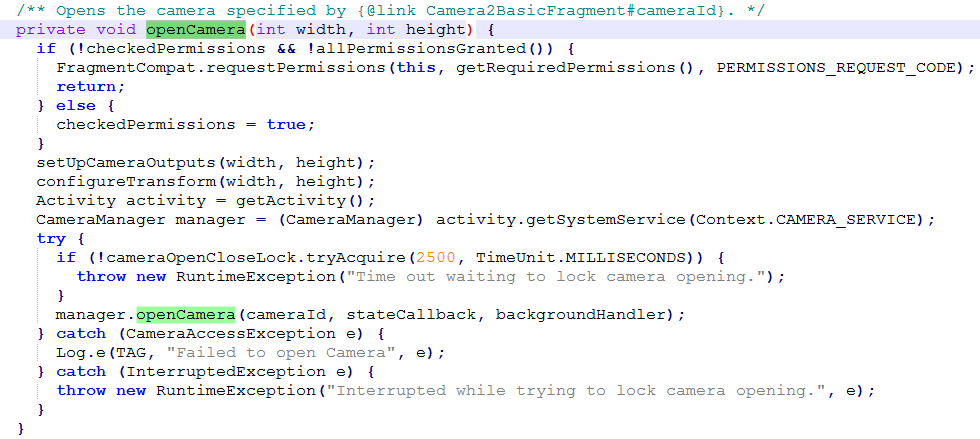
TextureView是显示图像的view.监听如下的事件,

1. onSurfaceTextureAvailable

会openCamera

1. onSurfaceTextureSizeChanged

openCamera会利用CameraManager实现.

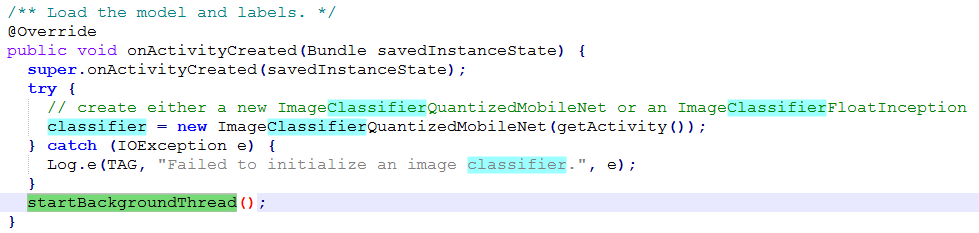


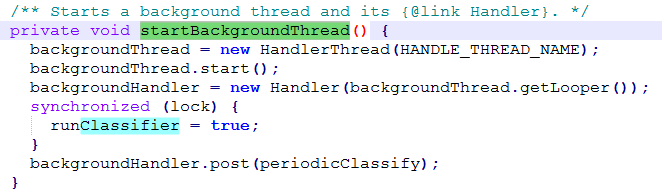
为了获得预览图像需要经过如下的过程：open CameraDevice, 提交 session, 接收结果：cameraDevice stateCallback -> create session -> session result：这个过程需要把texture和camera预览返回的image关联。

这里用到了上面提到的CameraDevice.StateCallback stateCallback以及createCameraPreviewSession(处理session以及发送preview的request).

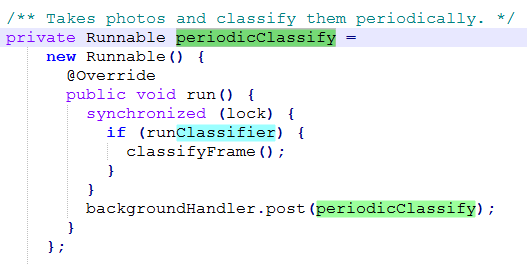
**BackgroundThread**

通过一个HandleThread周期性处理得到的frame，使用lock控制原子操作：classifyFrame

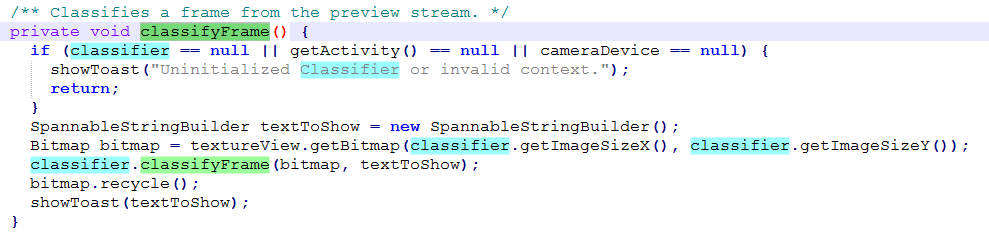




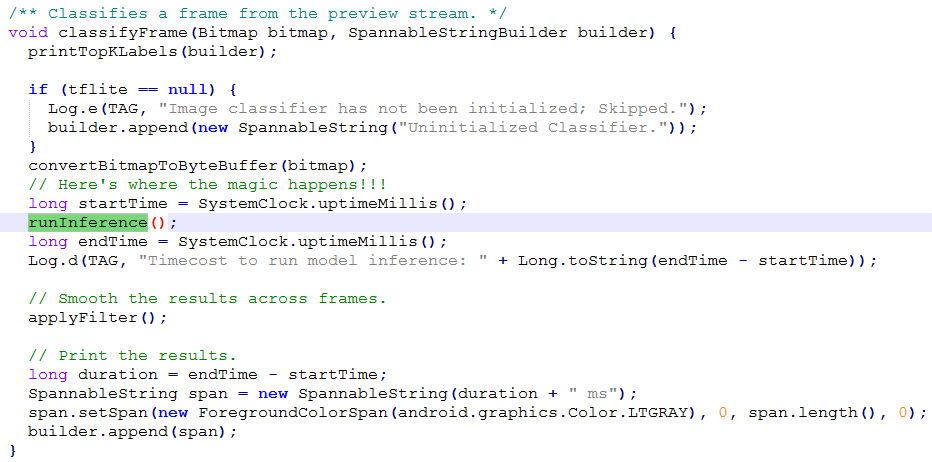
Backthread执行classifyFrame.



在classifyFrame中**做tflite的识别.**



ImageClassify.java



runInference是和tflite接口的.利用tflite.run.这部分其实换成其他的人脸,语义分割都不要改动. labelProbArray是网络的推断结果.

