Summary of Competition

(20181224)

Abstract: 这是最后一篇关于这次Arm比赛的文章，作为总结，我想解释一下之前的几篇文章的思路，还要补充一下使用Tengine过程中遇到的问题，最后讲讲这个比赛感觉上还有的不足之处和一些结果吧。

一、之前文章的思路：

实际上之前几篇文章都是按照这个比赛的完成顺序来的，在03当中讲述了整个Tengine\_FaceDetector的框架，这个框架的顺序也正好是按照这个比赛的完成顺序与文章的顺序来的：

（1）首先你要做安卓肯定还是要用到Android Studio的吧，所以00讲述了在安装Android Studio上的一些问题；

（2）然后尝试用Tengine在Android上的部署，所以有01这篇文章，但是用的模型是不一样的，是物体分类的模型，并没有用到人脸检测；

（3）有一天队友告诉我人脸检测要用到OpenCV库，于是又学习了一下OpenCV在Android上面的装载，所以有02这篇文章讲述了Android\_OpenCV

（4）最后我们将之前的总和起来就实现了03\_Tengine\_FaceDetector

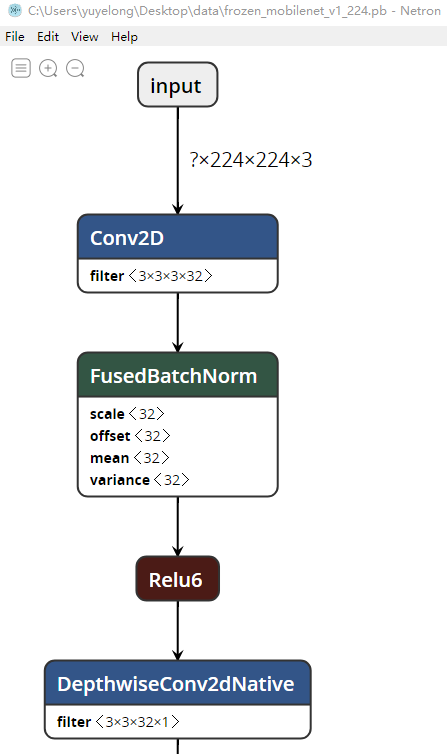
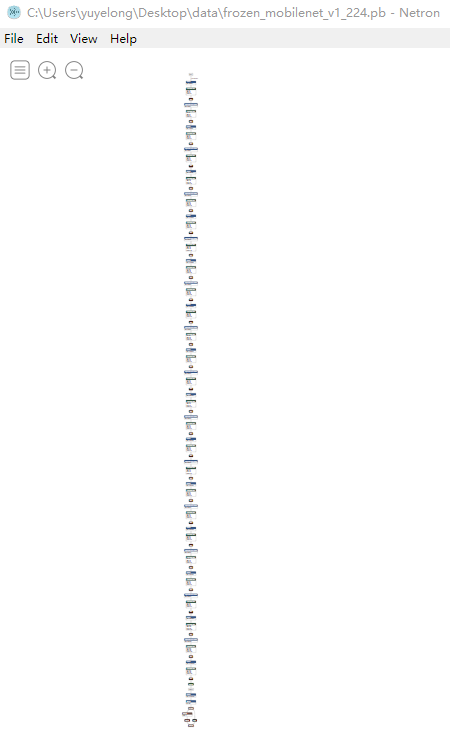
1. 补充使用Tengine过程中遇到的问题

1、模型load不成功

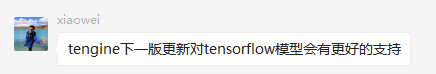
一开始load模型的时候发现不成功，究其原因原来是模型里面有些函数还不支持，tips和Github上有一些框架所支持的OP列表，除此之外还有一些函数虽然没有列出但是Tengine还是支持的，比如tensorflow的Reshape函数。

这里也介绍一个查看模型结构的利器：

<https://github.com/lutzroeder/netron>

通过这个利器就可以看到模型里面用到的函数，反复对比之后可以找到那些不支持的函数。现在的Tengine还在不断的完善中，据说tengine下一版更新对tensorflow模型会有更好的支持：



注：百度网盘里下载了许多Tengine所用的模型，里面有一些tensorflow模型，通过查看一些模型的结构就可以基本了解哪些没列出来的函数也是支持的。总之模型加载不进去的最大原因就是模型当中有些函数还不支持，所以需要改变结构，像我们的模型结构就改的很简单。

1. Mat函数的读取与赋值

这里不可避免地要用到Mat类型的数据，因为使用了OpenCV的库，下面的文章给出了Mat数据的读取与赋值：

<https://blog.csdn.net/u011028345/article/details/74852501>

但是注意的是这都是在C++/C环境下的读取方式，如果在java层的话读取方式有所不同，java层mat类内置了许多函数可以获取mat数据，比如：

Mat mygraypic.get(0,0)[0]

这里再说说mat数据类型的格式吧，类似矩阵，但是每行每列矩阵上的元素可能有多个通道RGB，所以get（row,col）之后是返回一个数组double[]，这个数组就包含了各个通道的值。

1. 数组数据的传递

**重新学习指针：数组指针、指针数组的概念，还有函数指针和指针函数。int \*p[] 与int (\*p)[]的不同，前者是定义一个数组，这个数组每个元素是指针；后者是定义一个指针，这个指针指向一个数组。**

现在要求是这样的，因为最后输出的一个数组，所以我们需要定义一个函数输出是数组的形式，而且在这里我们知道数组是30个元素，相对比较简单，所以我们直接申明一个输出为30个元素的数组：

**float** (\*TengineWrapper::FaceDetect())[30]  
{  
 **float** \*data = (**float** \*)get\_tensor\_buffer(output\_tensor);  
  
 **float** (\*datatmp)[30]=(**float** (\*)[30])**new float**[30];  
 **for**(**int** i=0;i<30;i++)  
 {  
 (\*datatmp)[i]=data[i];  
 //(\*datatmp)[i]=i+0.1;  
 }  
 **delete** datatmp;  
 **return** datatmp;  
}

在JNI层调用，定义输出为jfloatArray：

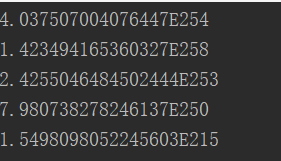
JNIEXPORT jfloatArray JNICALL  
Java\_com\_tengine\_openailab\_mobilenet\_MainActivity\_TengineWrapperFaceDetect(  
 JNIEnv \*env,  
 jobject /\* this \*/) {  
  
 //1.新建长度len数组  
 jfloatArray jarr = env->NewFloatArray(30);  
  
 //2.获取数组指针  
 jfloat \*arr = env->GetFloatArrayElements(jarr, NULL);  
 //3.赋值  
 **int** i = 0;  
 **float** (\*tmp)[30];  
 tmp=tengine\_wrapper->FaceDetect();  
 **for**(; i < 30; i++){  
 //arr[i] = \*(tmp+i);  
 arr[i] = (\*tmp)[i];  
 //arr[i] = i+0.1;  
 }  
 //4.释放资源  
 env->ReleaseFloatArrayElements(jarr, arr, 0);  
 //5.返回数组  
 **return** jarr;  
}

在java层调用，定义输出为float[]:

public native float[] TengineWrapperFaceDetect();

4、float和double数据不同

这个大bug是最后一段时间困扰多时的问题，因为之前用的是double数据类型，输入输出都是，但是导致输入的时候是double类型的小数，但是输出的数据超级大，大到离谱：



够离谱的吧，这个是科学计数法后面竟然是10的250多次方，于是找原因咯，首先要确定输入的数据是0到1之间的小数，看过了没问题，后来折腾了两天。最后想试试单张照片的时候，想着试试float类型的数据吧，结果发现可以了！得知模型里面的输入是float64的格式，于是寻找float32、float64、double数据的不同之处。所以以后的数据类型还是得严格一致才行啊！

再来看看mat数据中有这么一项CV\_32FC1，其中32F表示浮点数32位，C1表示通道数为1，类似地还有其他类型含义：

<https://blog.csdn.net/zssyu0416/article/details/82852339>

实验证明了只有CV\_32FC1是正确的在这个项目里，改成64也不行，还有一个是buffer里面申请的数据size，float是每个4位，double应该是需要8位。

1. 其他问题
2. 调试的过程——Log使用、return使用

Android Studio的调试主要依赖于Log日志的使用，《第一行代码》当中的1.4节讲述了Log日志的使用方式和类型。但是这里有一个问题，Log日志的调试只能停留在java层，对于底下的C++/C底层代码无能为力，于是这就会有一个问题是底下的代码我们无法调试。搜索了网上的信息之后发现Android Studio也可以调试底层的C++/C代码，但是我还没有试过。这里讲述一下我使用的比较笨的调试方法——多用return。

想要底层的数据传到java层只有一种方式，return数据，所以我们可以将处理过的数据经过return返回到顶层。打印的Log日志信息可以通过搜索找到特定的一句，再点击时间找到程序运行的时间点可以查看上下的信息。

1. 程序在手机上运行闪退

这个问题也是困扰多次的情况，其实这个时候我们也可以使用Log日志查看程序能够运行到哪里，因为出现闪退的地方后面的程序是运行不下去的，会出现红色报错。通过这种方式可以查看程序闪退的地方在哪里，然后才是去解决，比如出现下面红色的错误提示：

**Fatal signal 11 (SIGSEGV), code 1, fault addr 0xc91303b0 in tid 3587 (RenderThread)**

下面可以说明一下出现这种闪退的原因，列出一些我遇到过的出现闪退的原因：

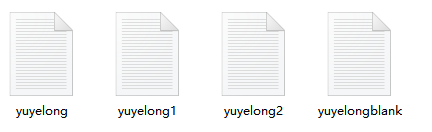
1. load不成功，原因是模型或者有些图片、文件在手机上没找到
2. 指针问题，超出了数组的范围
3. 其他想不起来了......

总之出现闪退肯定是你的程序出现了问题，反复查找错误就行了

1. 数据输出到文件，matlab

还有一种调试的方式是选择将数据直接输出到外面文件，因为底层是C++/C语言，所以按照C对文件的处理是可以执行的，但是这里的问题是这个文件需要先以一个空白文档形式adb传送到手机上。然后数据传到该文档之后再使用adb pull传出来之电脑。

我在这个工程里面就用过这种方法将一张照片的灰度图数据传出到txt文档，再用adb pull传出到电脑，然后将数据传送到Matlab里面显示图像：

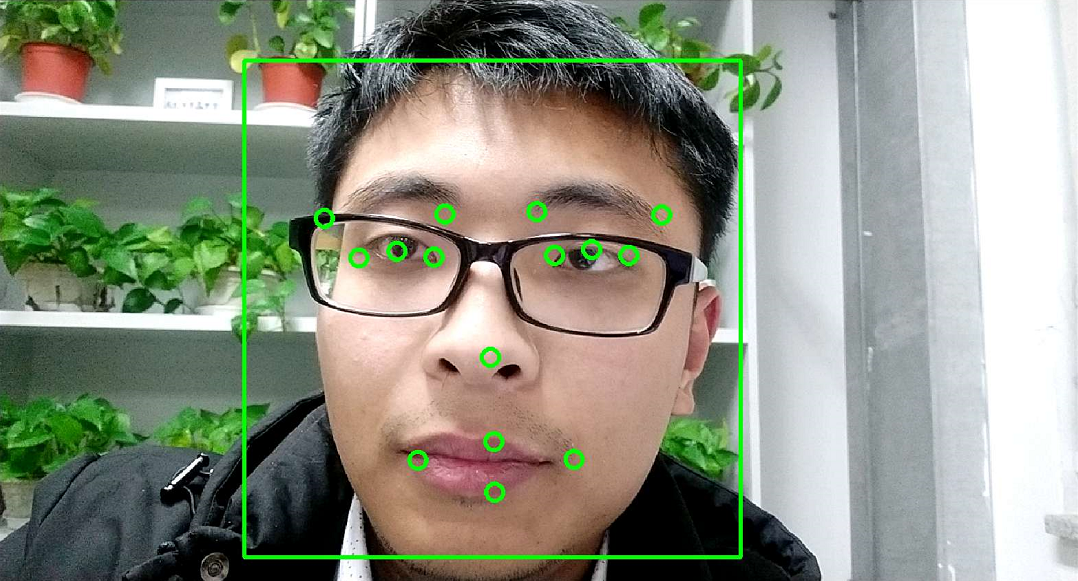


1. CV里面的坐标轴千万要注意，横轴和竖轴是反的！反的！这个你把照片调出来看看就应该知道了。Mat(x,y)，有些时候要读取一定范围的mat，应该是mat.rowrange(y).colrange(x)，例如：

mygraypic = new Mat();  
mygraypic = mGray.rowRange(y1, y2).colRange(x1, x2);

1. 总结

这个比赛总共历时4个礼拜的零零碎碎的时间，要感谢OpenAILab提供的很多资源，也要感谢公司技术的支持，还有俱乐部各位小伙伴答疑解惑，还有队友的友情帮助！在这个项目中我所做的基本是Android的部署，对模型的训练还不是很清楚，这也是接下去要学习的方向。最后的最后，上一张结果照片：



当然还有误识别的时候，原因在于OpenCV自带的分类器的效果不大好：

