电脑端代码--computerdetect.py

实现步骤:

1.利用剪刀石头布数据集和卷积神经网络训练出rps.h5模型，代码--traincode.ipynb

2.将h5模型转换为tflite模型，用于部署，并且将输出和输入层找出，用于android模型的输入输出，代码--convert.ipynb

3.部署到android 代码--TFLiteClassification,改造自Doi技术团队的方案

首先要在build.gradle导入两个库。

implementation 'org.tensorflow:tensorflow-lite:2.3.0'

implementation 'org.tensorflow:tensorflow-lite-support:0.1.0-rc1'

复制转换的预测模型到app/src/main/assets目录下，还有类别的标签，每一行对应一个标签名称。

private static final float[] IMAGE\_MEAN = new float[]{128.0f, 128.0f, 128.0f};

private static final float[] IMAGE\_STD = new float[]{128.0f, 128.0f, 128.0f};

public TFLiteClassificationUtil(String modelPath) throws Exception {

File file = new File(modelPath);

if (!file.exists()) {

throw new Exception("model file is not exists!");

}

try {

Interpreter.Options options = new Interpreter.Options();

// 使用多线程预测

options.setNumThreads(NUM\_THREADS);

// 使用Android自带的API或者GPU加速

NnApiDelegate delegate = new NnApiDelegate();

// GpuDelegate delegate = new GpuDelegate();

options.addDelegate(delegate);

tflite = new Interpreter(file, options);

// 获取输入，shape为{1, height, width, 3}

int[] imageShape = tflite.getInputTensor(tflite.getInputIndex("input\_1")).shape();

DataType imageDataType = tflite.getInputTensor(tflite.getInputIndex("input\_1")).dataType();

inputImageBuffer = new TensorImage(imageDataType);

// 获取输入，shape为{1, NUM\_CLASSES}

int[] probabilityShape = tflite.getOutputTensor(tflite.getOutputIndex("Identity")).shape();

DataType probabilityDataType = tflite.getOutputTensor(tflite.getOutputIndex("Identity")).dataType();

outputProbabilityBuffer = TensorBuffer.createFixedSize(probabilityShape, probabilityDataType);

// 添加图像预处理方式

imageProcessor = new ImageProcessor.Builder()

.add(new ResizeOp(imageShape[1], imageShape[2], ResizeOp.ResizeMethod.NEAREST\_NEIGHBOR))

.add(new NormalizeOp(IMAGE\_MEAN, IMAGE\_STD))

.build();

} catch (Exception e) {

e.printStackTrace();

throw new Exception("load model fail!");

}

}

为了兼容图片路径和Bitmap格式的图片预测，这里创建了两个重载方法，它们都是通过调用predict()

public int predictImage(String image\_path) throws Exception {

if (!new File(image\_path).exists()) {

throw new Exception("image file is not exists!");

}

FileInputStream fis = new FileInputStream(image\_path);

Bitmap bitmap = BitmapFactory.decodeStream(fis);

int result = predictImage(bitmap);

if (bitmap.isRecycled()) {

bitmap.recycle();

}

return result;

}

public int predictImage(Bitmap bitmap) throws Exception {

return predict(bitmap);

}

这里创建一个获取最大概率值，并把下标返回的方法，其实就是获取概率最大的预测标签。

public static int getMaxResult(float[] result) {

float probability = 0;

int r = 0;

for (int i = 0; i < result.length; i++) {

if (probability < result[i]) {

probability = result[i];

r = i;

}

}

return r;

}

这个方法就是Tensorflow Lite执行预测的最后一步，通过执行tflite.run()对输入的数据进行预测并得到预测结果，通过解析获取到最大的概率的预测标签，并返回。到这里Tensorflow Lite的工具就完成了。

private int predict(Bitmap bmp) throws Exception {

inputImageBuffer = loadImage(bmp);

try {

tflite.run(inputImageBuffer.getBuffer(), outputProbabilityBuffer.getBuffer().rewind());

} catch (Exception e) {

throw new Exception("predict image fail! log:" + e);

}

float[] results = outputProbabilityBuffer.getFloatArray();

Log.d(TAG, Arrays.toString(results));

return getMaxResult(results);

}

实时预测

使用相机实时预测并显示预测结果。以下为activity\_main.xml的代码

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

tools:context=".MainActivity">

<ImageView

android:id="@+id/image\_view"

android:layout\_width="match\_parent"

android:layout\_height="400dp" />

<TextView

android:id="@+id/result\_text"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@id/image\_view"

android:text="识别结果"

android:textSize="16sp" />

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:orientation="horizontal">

<Button

android:id="@+id/open\_camera"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="实时预测" />

</LinearLayout>

</RelativeLayout>

在MainActivity.java中，进入到页面我们就要先加载模型，我们是把模型放在Android项目的assets目录的，但是Tensorflow Lite并不建议直接在assets读取模型，所以我们需要把模型复制到一个缓存目录，然后再从缓存目录加载模型，同时还有读取标签名，标签名称按照训练的label顺序存放在assets的.txt，以下为实现代码。

classNames = Utils.ReadListFromFile(getAssets(), "label\_list.txt");

String classificationModelPath = getCacheDir().getAbsolutePath() + File.separator + "mobilenet\_v2.tflite";

Utils.copyFileFromAsset(MainActivity.this, "mobilenet\_v2.tflite", classificationModelPath);

try {

tfLiteClassificationUtil = new TFLiteClassificationUtil(classificationModelPath);

Toast.makeText(MainActivity.this, "模型加载成功！", Toast.LENGTH\_SHORT).show();

} catch (Exception e) {

Toast.makeText(MainActivity.this, "模型加载失败！", Toast.LENGTH\_SHORT).show();

e.printStackTrace();

finish();

}

摄像头实时预测

核心代码如下，创建一个子线程，子线程中不断从摄像头预览的AutoFitTextureView上获取图像，并执行预测，并在页面上显示预测的标签、对应标签的名称、概率值和预测时间。每一次预测完成之后都立即获取图片继续预测，只要预测速度够快，就可以看成实时预测。

private Runnable periodicClassify =

new Runnable() {

@Override

public void run() {

synchronized (lock) {

if (runClassifier) {

// 开始预测前要判断相机是否已经准备好

if (getApplicationContext() != null && mCameraDevice != null && tfLiteClassificationUtil != null) {

predict();

}

}

}

if (mInferThread != null && mInferHandler != null && mCaptureHandler != null && mCaptureThread != null) {

mInferHandler.post(periodicClassify);

}

}

};

// 预测相机捕获的图像

private void predict() {

// 获取相机捕获的图像

Bitmap bitmap = mTextureView.getBitmap();

try {

// 预测图像

long start = System.currentTimeMillis();

float[] result = tfLiteClassificationUtil.predictImage(bitmap);

long end = System.currentTimeMillis();

String show\_text = "预测结果标签：" + (int) result[0] +

"\n名称：" + classNames[(int) result[0]] +

"\n概率：" + result[1] +

"\n时间：" + (end - start) + "ms";

textView.setText(show\_text);

} catch (Exception e) {

e.printStackTrace();

}

}

本项目中使用的打开相机的权限，要在AndroidManifest.xml添加以下权限申请。

<uses-permission android:name="android.permission.CAMERA"/>

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"/>

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

如果是Android 6 以上的设备还要动态申请权限。

// check had permission

private boolean hasPermission() {

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {

return checkSelfPermission(Manifest.permission.CAMERA) == PackageManager.PERMISSION\_GRANTED &&

checkSelfPermission(Manifest.permission.READ\_EXTERNAL\_STORAGE) == PackageManager.PERMISSION\_GRANTED &&

checkSelfPermission(Manifest.permission.WRITE\_EXTERNAL\_STORAGE) == PackageManager.PERMISSION\_GRANTED;

} else {

return true;

}

}

// request permission

private void requestPermission() {

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {

requestPermissions(new String[]{Manifest.permission.CAMERA,

Manifest.permission.READ\_EXTERNAL\_STORAGE,

Manifest.permission.WRITE\_EXTERNAL\_STORAGE}, 1);

}

}