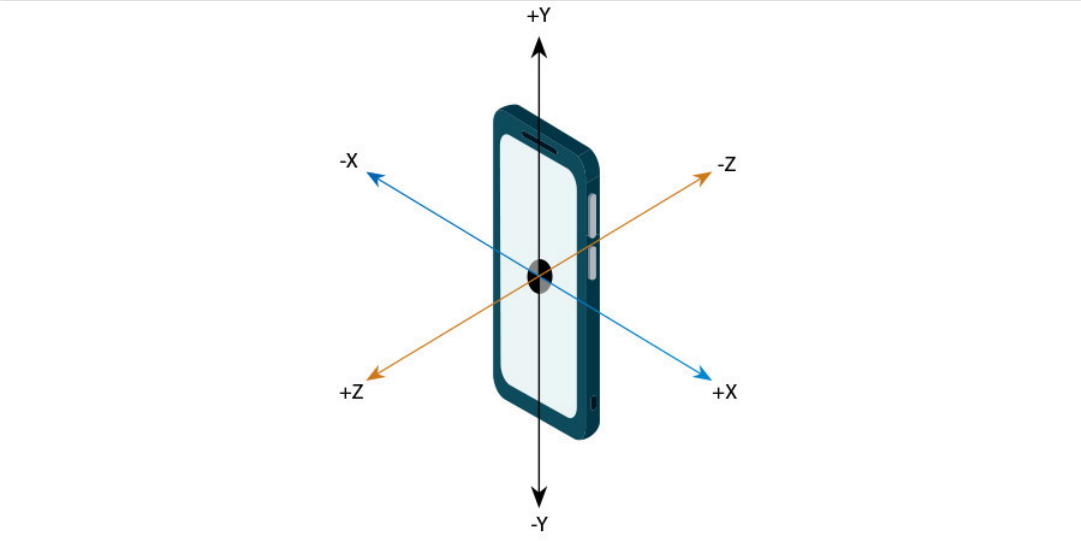
企鹅号小编 发表于 人工智能

安卓手机如何玩转“动作手势检测”？有TensorFlow就够了



原文来源：Lemberg Solutions Ltd

作者：Zahra Mahoor、Jack Felag、 Josh Bongard

「雷克世界」编译：嗯~阿童木呀、KABUDA

现如今，与智能手机进行交互的方式有很多种：触摸屏、硬件按钮、指纹传感器、视频摄像头（如人脸识别）、方向键（D-PAD）、手持设备控制等等。但是我们该如何使用动作识别功能呢?

我们可以举一个例子来说明这个问题，比如当你持手机将其快速移动到左侧或右侧时，可以非常精确地显示出想要切换到播放列表中下一首或上一首歌曲的意图；或者，你可以将手机快速向上向下翻转，从而刷新应用程序内容。引入这样的交互看起来是非常有发展前景的，并且为用户体验增添了一个新的层面。接下来，本文将介绍该如何使用机器学习和Android上的Tensorflow库实现这一目标。

对于我们的目标，我们可以将其描述为希望手机能够识别左右的快速动作。

我们希望能够在一个单独的Android库中完成这一实现，以便它能够容易地集成到任何其他应用程序中。

这些动作可以通过手机上的几个传感器进行捕获：加速度计、陀螺仪、磁力计等等。随后，这些批量动作可以用于机器学习算法，以便进行训练和后续识别。

为了捕捉数据，我们将开发一个Android应用程序。预处理和训练过程将在Jupyter Notebook环境的PC上使用Python和TensorFlow库执行。手势识别将在一个Android应用程序演示中执行，并生成训练数据。最后，我们将开发一个即时可用的Android库，用于手势识别，而且可以很容易地集成到其他应用程序中。

下面是有关该实现过程的高级计划：

3.将神经网络导出到手机中

4.开发一个测试Android应用程序

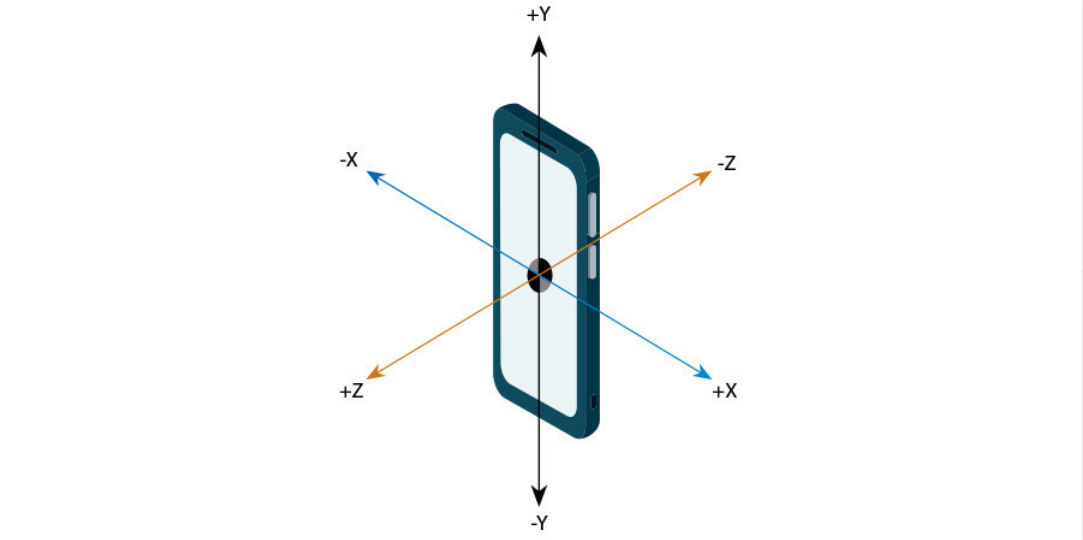
5.开发Android库

实现

・准备数据

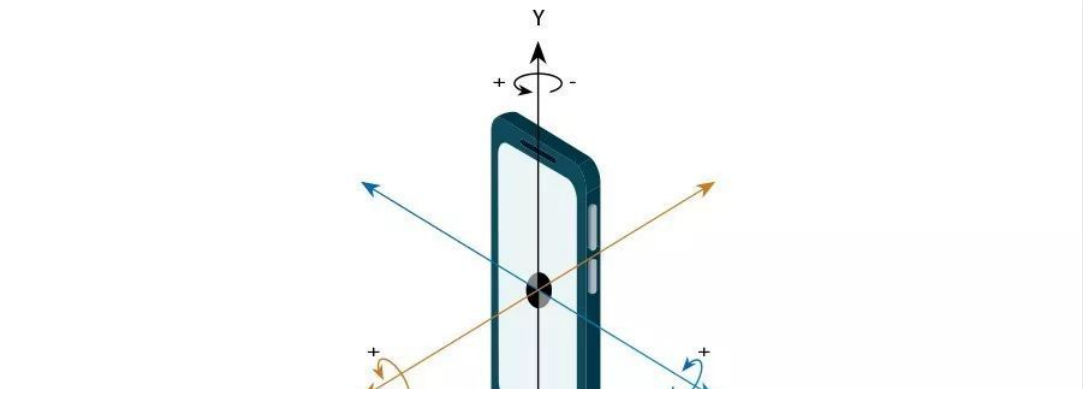
首先，我们来明确一下什么样的传感器和什么样的数据可以用于描述我们的动作手势。为了准确地描述这些手势，我们应该使用加速度计和陀螺仪。

加速度计传感器明显是用于测量加速度，然后测量运动：



加速度计有一个有趣的细微差别；它不仅测量设备本身的加速度，而且测量地球重力的加速度，大约为9.8m/s2。这意味着放在桌子上的手机加速度向量的大小将等于9.8。这些值不能直接使用，而应从地球重力值中提取。这并不是一件容易的任务，因为它需要磁力计和加速度计传感器值的融合。不过幸运的是，Android已经有了特定的线性加速度传感器以执行这样的计算并返回正确的值。

另一方面，陀螺仪将用于测量旋转：

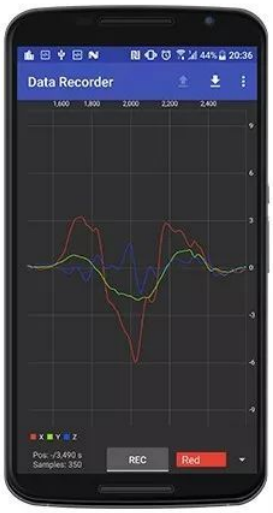


我们试着找出将与我们的手势相关联的值。显然，在加速度计（即线性加速度计）中，X和Y值将高度描述手势，而加速度计的Z值不太可能受到我们手势的影响。

至于陀螺仪传感器，似乎只有Z轴会受到手势的轻微影响。不过，为了简化实施，我们可以不将该值考虑在内。在这种情况下，我们的手势检测器不仅能够识别手机在手中的移动，而且还能识别出其沿着水平线的移动——例如在桌子上。这大概不是一个太大的问题。

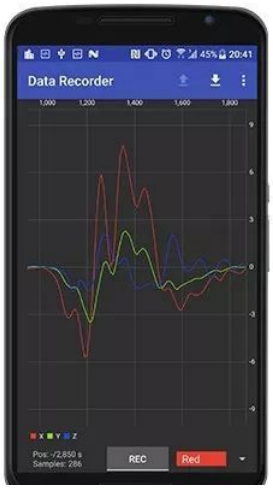
所以，我们需要开发一个能够记录加速度计数据的Android应用程序。

我开发了一款这样的应用程序，下面是记录的“向右移动”手势的屏幕截图：



正如你所看到的那样，X轴和Y轴对手势的反应非常强烈。 Z轴也有反应，但正如我们所说的那样，我们没有将其考虑在内。

这是“左移”手势：



有任意，x值与前一个手势的值允许相反。

还有一点需要提及的是数据采样率。这反映了数据采样的频率，并且直接影响每个时间间隔的数据量。

另一个要考虑的是手势持续时间。这个值，就像这里许多其他值一样，应该根据经验对其进行选择。我所建立的那个手势持续时间不超过1秒，但为了让事情进行得更为顺利，我把它四舍五入到了1.28秒。

我选择的数据采样率是每个选定的持续时间内128点，这将产生10毫秒的延迟（1.28 / 128）。这个值应该被传递给registerListener（

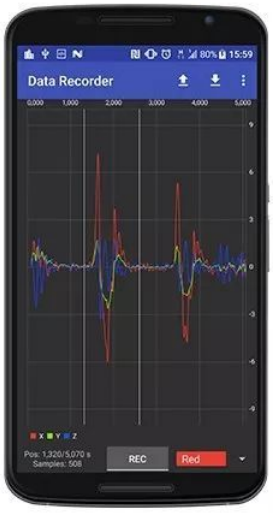
https://developer.android.com/reference/android/hardware/SensorManager.html#registerListener%28android.hardware.SensorEventListener,%20android.hardware.Sensor,%20int%29

）方法。

因此，这个想法就是训练一个神经网络，用以在加速度传感器中的实时数据流中识别这些信号。

所以，接下来，我们需要记录一系列手势并将其导出到文件。当然，相同类型的手势（右侧或左侧）应该使用相同的标签进行标记。我们很难事先说出需要多少样本来训练这个网络，但这可以通过训练结果进行确定。

通过点击数据，样本持续时间将自动高亮显示：



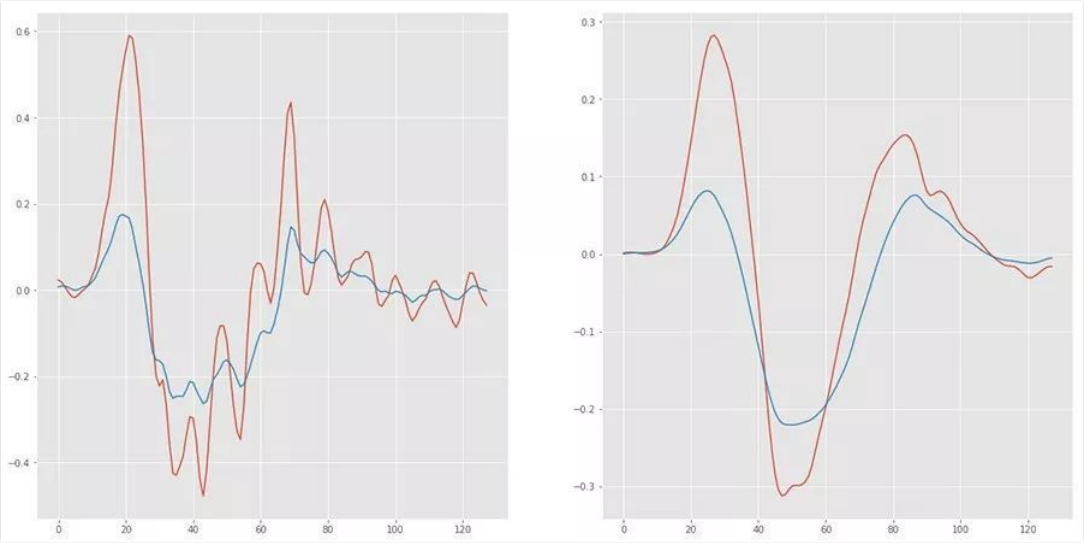
现在，“保存”按钮启用，点击它将自动把已选项保存到工作目录中的文件。文件名将以“ _ .log”的形式生成。可以使用应用程序菜单来选择工作目录。

另外请注意，保存当前选择后，将自动选择下一个手势。下一个手势的选择是使用一个非常简单的算法实现的：找到其绝对值大于3的第一个X条目，然后返回20个样本。

这种自动化过程使我们能够快速保存一系列样本。对于每个手势我们记录了500个样本。保存的数据应该被复制到一台PC上以进行进一步处理。 （直接在手机上进行处理和训练看起来很有发展前景，但是Android的TensorFlow目前不支持训练）。

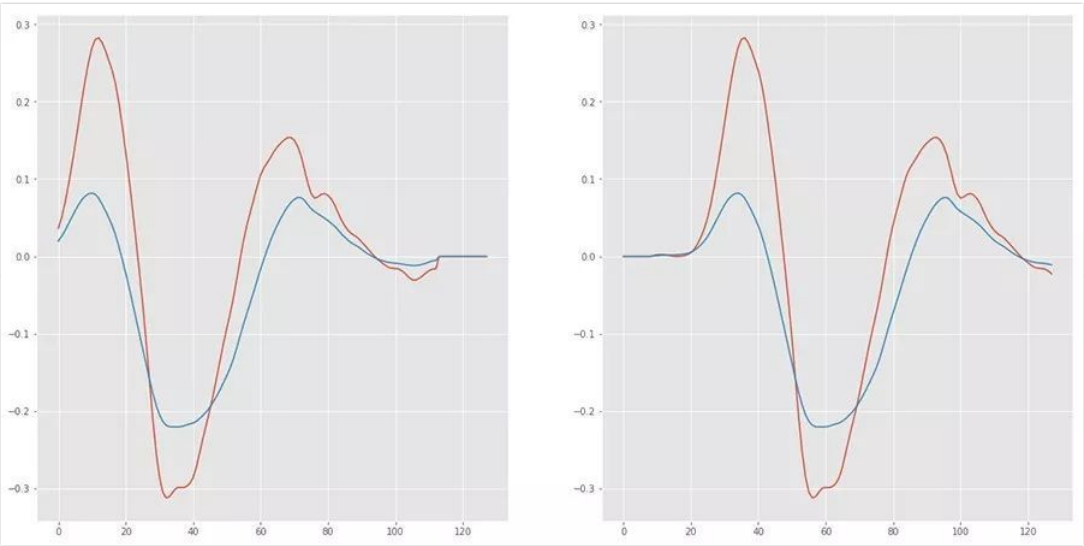
在前面提供的截图中，数据范围大约为±6。但是，如果你更有力地挥动手机，它可以达到±10。对数据进行正则化操作比较好，从而使得范围为±1，这更适合神经网络数据格式。我只是把所有的数据除以一个常数系数，我通常使用

有很多方法可以过滤数据。一种是基于移动平均值（https://en.wikipedia.org/wiki/Moving_average）框进行过滤。



请注意，X数据的最大值现在是原来值的一半。由于我们将在识别过程中对实时数据执行相同的过滤，所以这应该不成问题。

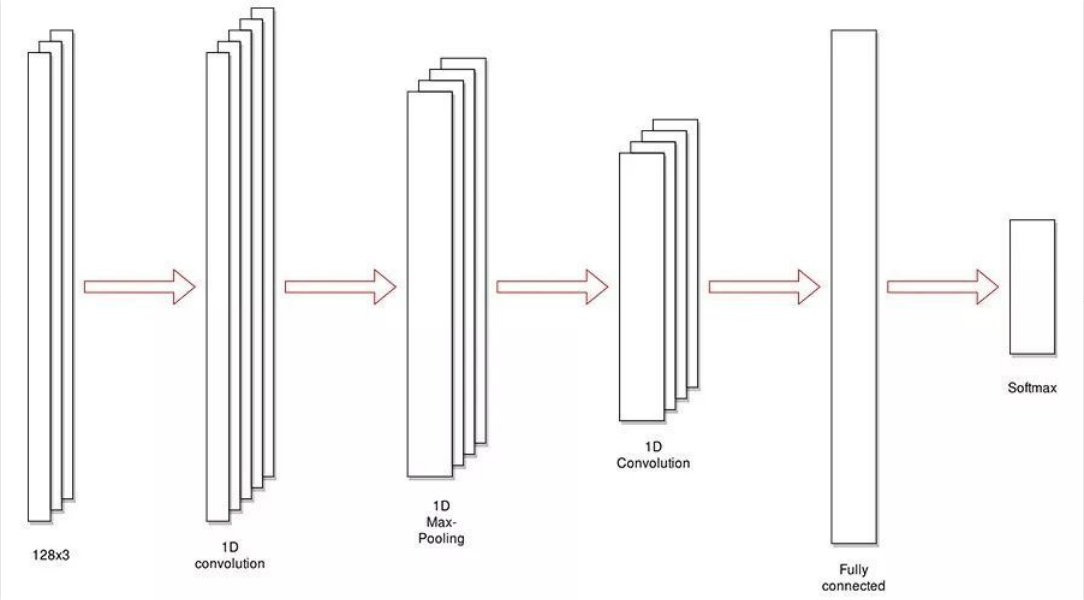
改善训练的最后一步是数据增强（data augmentation）。该过程通过执行一些操作扩展了原始数据集。在我们的例子中，我简单地将数据左右移动了几个点：



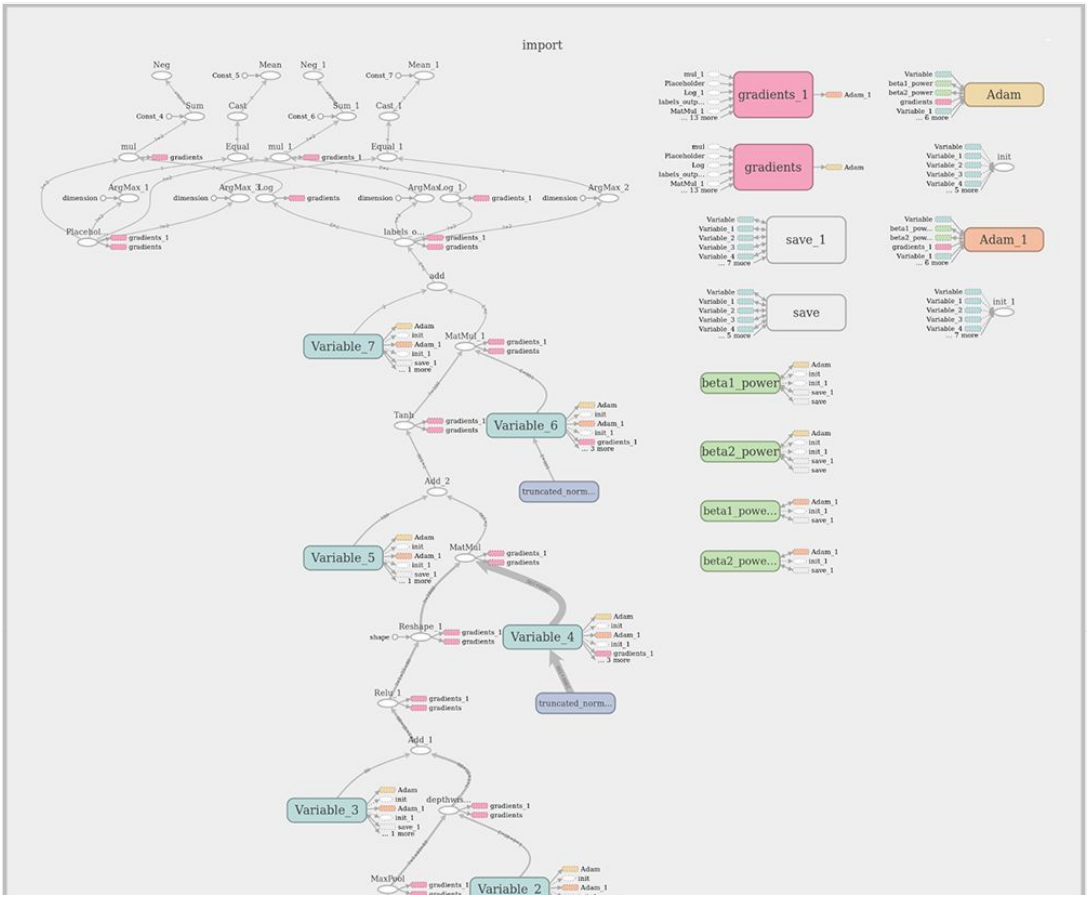
・设计一个神经网络

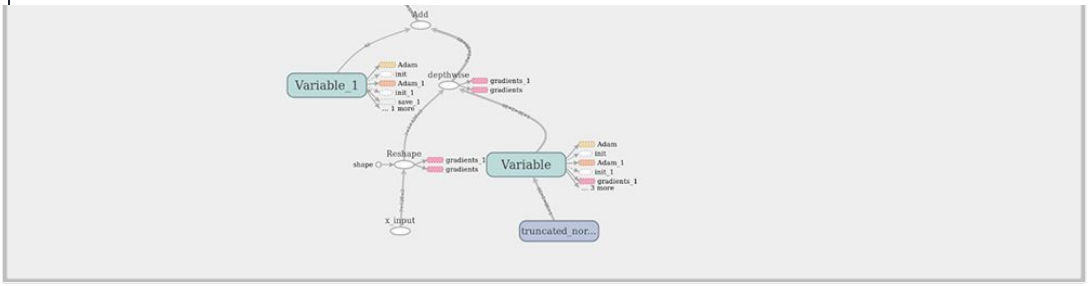
设计一个神经网络并不是一个简单的任务，需要一些经验和直觉。另一方面，神经网络在某些任务中是众所周知的，而且你可以简单地对现有网络做些调整。我们的任务与图像分类任务非常相似，输入可以被视为高度为1像素的图像（这是真实的——第一个操作是将二维数据[128列x 2信道]的输入转换为三维数据[1行x 128列x 2信道]）。

这里是神经网络的示意图：



这里是通过TensorBoard获得的详细原理图：





该示意图包含一些仅用于训练的辅助节点。之后，我将提供一个干净的、优化后的图片。

・训练

训练将在具有Jupyter Notebook环境的PC上使用Python和TensorFlow库进行。可以使用以下配置文件在Conda环境中启动Notebook。以下是一些训练超参数（hyperparameters）：

Optimizer: Adam

Number of training epochs: 3

Learning rate: 0.0001

数据集按照7/3的比例被分为训练集和验证集。

训练质量可以通过训练和测试精确度值来控制。训练的精确度应该接近但不能达到1。如果值太低，则表示识别率低且不准确，并且过高的值将导致模型过度拟合，并且可能在识别期间引入一些伪像，如非手势数据的非零识别评估。良好的测试精度可以证明：一个训练有素的模型可以识别不可见的早期数据。

训练日志：

TensorFlow图和相关数据使用以下方法保存到文件中：

```
saver = tf.train.Saver()

with tf.Session() as session:

session.run(tf.global_variables_initializer())

# save the graph

for epoch in range(training_epochs):

# train

saver.save(session, './session.ckpt')
```

完整的notebook代码可以在这里查看：

・输出神经网络

上一部分展示了如何保存Tensor Flow数据。图形保存到“session.pb”文件中，训练数据（权重，偏差等）被保存到一批“session.ckpt”文件中。这些文件可以足够大：

session.pb 47732

命令：

```
python freeze_graph.py --input_graph=session.pb \

--input_binary=True \

--input_checkpoint=session.ckpt \

--output_graph=frozen.pb \

--output_node_names=labels_output
```

生成的文件比之前的文件要小，但就单个文件而言，仍然足够大：

```
frozen.pb                                                    1130835
```

这是TensorBoard中的模型：

tensorflow / python / tools / import_pb_to_tensorboard.py文件复制到notebook目录并启动：

```
python import_pb_to_tensorboard.py --model_dir=frozen.pb --log_dir=tmp
```

其中frozen.pb是一个模型文件。

现在，启动TensorBoard：

```
tensorboard --logdir=tmp
```

有几种方法可以为移动环境优化模型。想要运行所描述的命令，需要从源代码编译TensorFlow：

1.删除未使用的节点和常规优化。执行：

```
bazel build tensorflow/tools/graph_transforms:transform_graph

bazel-bin/tensorflow/tools/graph_transforms/transform_graph --in_graph=mydata/frozen.pb
--out_graph=mydata/frozen_optimized.pb --inputs='x_input' --outputs='labels_output'
--transforms='strip_unused_nodes(type=float, shape="128,2") remove_nodes(op=Identity,
op=CheckNumerics) round_weights(num_steps=256) fold_constants(ignore_errors=true) fold_batch_norms
fold_old_batch_norms'
```

这是TensorBoard运行结果：

2.执行量化（转换浮点数据格式信息8位数据格式），执行：

```
bazel-bin/tensorflow/tools/graph_transforms/transform_graph --in_graph=mydata/frozen_optimized.pb
--out_graph=mydata/frozen_optimized_quant.pb --inputs='x_input' --outputs='labels_output'
--transforms='quantize_weights strip_unused_nodes'
```

因此，与3.5 Mb的原始文件相比，输出文件的大小为287129字节。这个文件可以在Android的TensorFlow中使用。

・演示 Android应用程序

想在Android应用程序中执行信号识别，你需要使用Android的Tensor Flow库。将库添加至gradle属性项：

```
}
```

现在，你可以通过TensorFlowInferenceInterface类访问TensorFlow API。首先，将“frozen_optimized_quant.pb”文件放入应用程序的“assets”目录中（即“app/src/main/assets”），并将其加载到代码中（如：从Activity开始；但是，像往常一样，最好在后台线程中执行有关IO的相关操作）

```
inferenceInterface = new TensorFlowInferenceInterface(getAssets(), “file:///android_asset/frozen_optimized_quant.pb”);
```

注意如何选定模型文件

最后，看一下如何进行识别：

```
float[] data = new float[128 * 2];

String[] labels = new String[]{"Right", "Left"};

float[] outputScores = new float[labels.length];

// populate data array with accelerometer data

inferenceInterface.feed("x_input", data, new long[] );

inferenceInterface.run(new String[]{“labels_output”});

inferenceInterface.fetch("labels_output", outputScores);
```

数据是我们“黑盒”的一个输入，应该有一个加速度计X和Y测量的平面阵列，即数据格式是[x1，y1，x2，y2，x3，y3，…，x128，y128]。

在输出方面，我们有两个浮点值，它们根据“左”或“右”的手势变化，随不同输入值在0到1的范围内取值。需要注意的是，这些值的总和是1。因此，举一个例子，如果输入信号与左或右手势不匹配，那么输出将接近[0.5,0.5]。为了进一步简化，最好使用简单的数学方法将这些值转化为绝对值在0到1之间的数。

在运行之前，需要对数据进行过滤和正则化操作。

这里是演示应用程序的最终测试屏幕：

其中“红色”和“绿色”的线条是实时预处理信号，黄线和青线分别属于“固定”、“左”和“右”概率。“时间”是处理时间，非常短，这使得实时识别成为可能。（2毫秒意味着可以以500Hz的频率运行处理，同时我们请求加速度计以100Hz的频率进行更新）。

正如你所看到的，有一些令人惊奇的细微差别。首先，即便是“沉默”信号，也存在一些非零概率。其次，每个手势在中心都具有长时间的“真实”识别，其值接近于1，并且在边缘处具备较小的相反识别。

看起来，要执行准确的实际手势识别，需要进行一些附加的处理。

・Android库

我在一个单独的Android库中对输出信号进行附加处理，实现了TensorFlow识别。以下是库和演示应用程序。

如果在你自己的应用程序中使用它，请将库属性项添加至模块gradle文件中：

```
repositories { maven { url "https://dl.bintray.com/rii/maven/" } }

dependencies {

...

}
```

创建一个动作检测器监听器（MotionDetector listener）：

```
private final MotionDetector.Listener gestureListener = new MotionDetector.Listener() {

@Override

public void onGestureRecognized(MotionDetector.GestureType gestureType) {

Log.d(TAG, "Gesture detected: " + gestureType);

}

};
```

启用动作检测：

```
MotionDetector motionDetector = new MotionDetector(context, gestureListener);

motionDetector.start();
```

我们通过利用TensorFlow库，在Android应用程序上实现了对动作手势进行识别的所有步骤：采集和预处理训练数据、设计和训练神经网络、开发测试应用程序以及随时可用的Android库。所描述的方法可以用于其他任何识别/分类任务。生成的库可以集成到其他任何Android应用程序中，并通过动作手势进行升级。

中国人工智能产业创新联盟于2017年6月21日成立，超200家成员共推AI发展，相关动态：

本文来自企鹅号 - 雷克世界媒体

发表于 2018-01-24

数据处理 Android TensorFlow

人工智能
507 篇文章 77 人订阅

订阅专栏

0 条评论

写下你的评论

评论

相关文章

来自专栏 人工智能LeadAI

配置深度学习主机与环境（TensorFlow+1080Ti） | 第四章 基于Anaconda的TensorFlow安装

60 5

来自专栏 人工智能LeadAI

译文 | 怎样用 JRebel for Android

50 4

来自专栏 生信技能树

RNA-seq老司机领读转录组结题报告

61 5

来自专栏 编码小白

cordova学习三 平台添加

57 9

来自专栏 人工智能LeadAI

配置深度学习主机与环境（TensorFlow+1080Ti） | 第二章 Win10&Ubuntu双系统与显卡驱…

71 6

来自专栏 人工智能LeadAI

TensorFlow从1到2 | 第一章 消失的梯度

61 5