TensorFlow Lite Python对象检测示例与Pi相机

这个例子使用TensorFlow Lite和Python在一个Raspberry Pi上执行实时对象检测，使用的图像流从Pi相机。它在摄像机预览中为每个检测到的对象绘制一个边界框(当对象的分数超过给定的阈值)。

虽然这里的TensorFlow模型和几乎所有代码都可以与其他硬件一起工作，但代码使用' picamera ' API从Pi相机捕捉图像。如果你想使用不同的摄像头输入，你可以修改代码的这些部分。

在本页的末尾，有一些额外的步骤来使用Coral USB加速器加速示例，它将推理速度提高了约10倍。

1.设置硬件

在你开始之前，你需要设置Pi

您还需要连接并配置Pi摄像头。

要想看到相机拍摄的结果，你需要一个连接树莓派的监视器。如果您正在使用SSH访问Pi shell(您不需要使用连接到Pi的键盘)，这是可以的——您只需要一个监视器连接到Pi就可以看到相机流。

2.安装TensorFlow Lite运行时

在这个项目中，你所需要的TensorFlow Lite API就是`Interpreter`类。
所以我们没有安装大的`tensorflow`包，而是使用小得多的`tflite_runtime`包。

要在Raspberry Pi上安装该文件，请按照Python quickstart中的说明操作。执行`pip install`命令后返回这里。

3.下载示例文件

首先，复制这个Git repo到你的树莓派，就像这样:

```
git clone https://github.com/tensorflow/examples --depth 1
```

然后使用我们的脚本安装一对Python包，并下载MobileNet模型和标签文件:

```
cd examples/lite/examples/object_detection/raspberry_pi

# 该脚本接受一个参数，指定您希望将模型文件保存在何处
bash download.sh /tmp
```

4.运行示例

```
python3 detect_picamera.py \
  --model /tmp/detect.tflite \
  --labels /tmp/coco_labels.txt
```

你应该会看到摄像头的信号出现在显示器上，连接到你的树莓派。在相机前放置一些物体，比如一个咖啡杯或键盘，你会看到在模型识别的物体周围画上方框，包括标签和每个物体的分数。它还在屏幕左上角以毫秒为单位打印执行每个推断所花费的时间。

有关使用TensorFlow Lite执行推断的更多信息，请阅读TensorFlow Lite推断。

5.文件说明

- Python包：numpy、picamera、Pillow、tflite_runtime
- 程序文件：detect_picamera.py、annotation.py

- 标签文件：coco_labels.txt
- 模型文件：detect.tflite
- 模型版本：coco_ssd_mobilenet_v1_1.0_quant_2018_06_29

6.加速推理时间(可选)

如果你想显著加快推理的时间,您可以附加一个毫升加速器等coral USB 加速器——USB附件,添加边缘TPU ML加速器到任何基于linux的系统。

如果你有一个coral USB加速器，遵循这些额外的步骤，委托模型执行的边缘TPU处理器:

1. 首先，确保你已经完成了USB加速器设置说明。
2. 现在打开`detect_picamera.py`文件，并在顶部添加以下导入:

   ```
   from tflite_runtime.interpreter import load_delegate
   ```

   然后找到初始化`Interpreter`的那一行，它看起来像这样:

   ```
   interpreter = Interpreter(args.model)
   ```

   更改为指定边缘TPU委托:

   ```
   interpreter = Interpreter(args.model,
       experimental_delegates=[load_delegate('libedgetpu.so.1.0')])
   ```

   `libedgetpu.so.1.0`文件是由你在步骤1中安装USB加速器时安装的Edge TPU库提供的。
3. 最后，您需要一个为Edge TPU编译的模型版本。

   通常，你需要使用Edge TPU编译器来编译你的`.tflite`文件。但是编译器工具与树莓派不兼容，所以我们在"下载"中包含了一个预编译版本的模型`download.sh`。

   所以你已经有了你需要的编译模型:
   `mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tflite`.

现在您已经准备好在边缘TPU上执行TensorFlow Lite模型了。
只是`detect_picamera.py`，但请确保您指定了为Edge TPU编译的模型(它使用相同的标签文件和以前一样):

```
python3 detect_picamera.py \
  --model /tmp/mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tflite \
  --labels /tmp/coco_labels.txt
```

您应该会看到明显更快的推理速度。

有关使用coral设备创建和运行TensorFlow Lite模型的更多信息，请阅读TensorFlow models on the Edge TPU。