（已開源）5行代碼，快速實現圖像分割，逐行詳解，手誘教你處理圖像

OpenCV與AI深度學習 2021-12-10 08:13

收錄於句子
#計算機視覺 2 #圖像處理 21

編者薦語

圖像分割作為計算機視覺的基礎，是圖像理解的重要組成部分，也是圖像處理的難點之一。

轉載自丨量子位

當然，那麼好用的項目，開源是必須的。

為什麼要到圖像分割?

計算機視覺研究助理，會經常接觸到圖像分割的問題，但我們還需要做下面的“詳述”（容易初級）。

我們都知道每個圖像都有一個像素值組成。簡單來說，圖像分割在像素級上，對圖像進行的任務。

圖像分割中使用的一些“獨門秘技”，可以解決一些關鍵的計算機視覺任務。主要分為2類：

- **每一個分類**：就是把圖像中賦予一個類別標籤，用不同的顏色來表示。
- **实例分割**：它不需要对每个像素进行标记，它只需要找到感兴趣物体的边缘轮廓就行。

它的身影也经常会出现在比较重要的场景中：

- 无人驾驶汽车视觉系统，可以有效的理解道路场景。
- 医疗图像分割，可以帮助医生进行诊断测试。
- 卫星图像分析，等等。

所以，图像分割技术的应用还是非常重要的。

接下来，我们就直奔主题，开始了解一下PixelLib，这个神奇又好用的库。

快速安装PixelLib

PixelLib这个库可以非常简单的实现图像分割——5行代码就可以实现语义分割和实例分割。

老规矩，先介绍一下**安装环境**。

安装最新版本的TensorFlow、Pillow、OpenCV-Python、scikit-image和PixelLib：

```
pip3 install tensorflow
pip3 install pillow
pip3 install opencv-python
pip3 install scikit-image
pip3 install pixellib
```

PixelLib实现语义分割

PixelLib在执行语义分割任务时，采用的是Deeplabv3+框架，以及在pascalvoc上预训练的Xception模型。

用在pascalvoc上预训练的Xception模型执行语义分割：

```
import pixellib
from pixellib.semantic import semantic_segmentation
segment_image = semantic_segmentation()
segment_image.load_pascalvoc_model("deeplabv3_xception_tf_dim_ordering_tf_kernels.
segment_image.segmentAsPascalvoc("path_to_image", output_image_name = "path_to_out
```

让我们看一下每行代码：

```
import pixellib
from pixellib.semantic import semantic_segmentation

#created an instance of semantic segmentation class
segment_image = semantic_segmentation()
```

用于执行语义分割的类，是从pixellib导入的，创建了一个类的实例。

```
segment_image.load_pascalvoc_model("deeplabv3_xception_tf_dim_ordering_tf_kernels.
```

调用函数来加载在pascal voc上训练的xception模型(xception模型可以从文末传送门链接处下载)。

```
segment_image.segmentAsPascalvoc("path_to_image", output_image_name = "path_to_out
```

这是对图像进行分割的代码行，这个函数包含了两个参数：

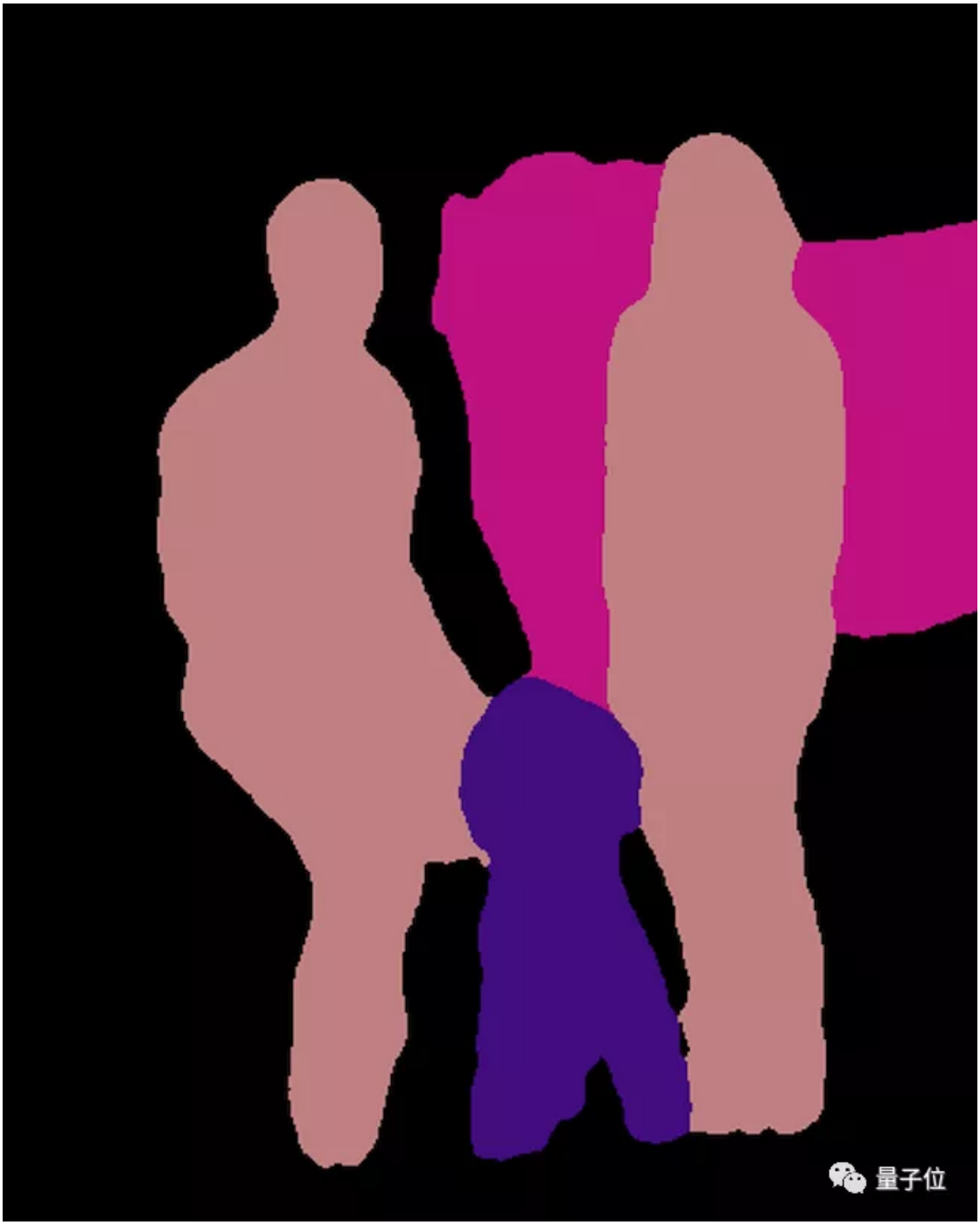
- path_to_image：图像被分割的路径。
- path_to_output_image：保存输出图像的路径，图像将被保存在你当前的工作目录中。

接下来，**上图，实战**！

图像文件命名为：sample1.jpg，如下图所示。

执行代码如下：

```
import pixellib
from pixellib.semantic import semantic_segmentation
segment_image = semantic_segmentation()
segment_image.load_pascalvoc_model("deeplabv3_xception_tf_dim_ordering_tf_kernels.
segment_image.segmentAsPascalvoc("sample1.jpg", output_image_name = "image_new.jpg
```



可以看到，在执行代码后，保存的图像中，所有对象都被分割了。

也可以对代码稍作修改，获取一张带有目标对象分割重叠(segmentation overlay)的图像。

```
segment_image.segmentAsPascalvoc("sample1.jpg", output_image_name = "image_new.jpg
```

添加了一个额外的参数，并设置为`True`，就生成了带有分隔叠加的图像。

可以通过修改下面的代码，来检查执行分割所需的推理时间。

```
import pixellib
from pixellib.semantic import semantic_segmentation
import time
segment_image = semantic_segmentation()
segment_image.load_pascalvoc_model(“pascal.h5”)
start = time.time()
segment_image.segmentAsPascalvoc(“sample1.jpg”, output_image_name= “image_new.jpg”
```

```
end = time.time()
print(f”Inference Time: {end-start:.2f}seconds”)
```

输出如下：

```
Inference Time: 8.19seconds
```

可以看到，在图像上执行语义分割，只用了8.19秒。

这个xception模型是用pascalvoc数据集训练的，有20个常用对象类别。

对象及其相应的color map如下所示：

PixelLib实现实例分割

虽然语义分割的结果看起来还不错，但在图像分割的某些特定任务上，可能就不太理想。

在语义分割中，相同类别的对象被赋予相同的colormap，因此语义分割可能无法提供特别充分的图像信息。

于是，便诞生了**实例分割**——同一类别的对象被赋予不同的colormap。

PixelLib在执行实例分割时，基于的框架是Mask RCNN，代码如下：

```
import pixellib
from pixellib.instance import instance_segmentation
segment_image = instance_segmentation()
segment_image.load_model("mask_rcnn_coco.h5")
segment_image.segmentImage("path_to_image", output_image_name = "output_image_path
```

同样，我们先来拆解一下每行代码。

```
import pixellib
from pixellib.instance import instance_segmentation
segment_image = instance_segmentation()
```

导入了用于执行实例分割的类，创建了该类的一个实例。

```
segment_image.load_model("mask_rcnn_coco.h5")
```

这是加载 Mask RCNN 模型来执行实例分割的代码(Mask RCNN模型可以从文末传送门链接处下载)。

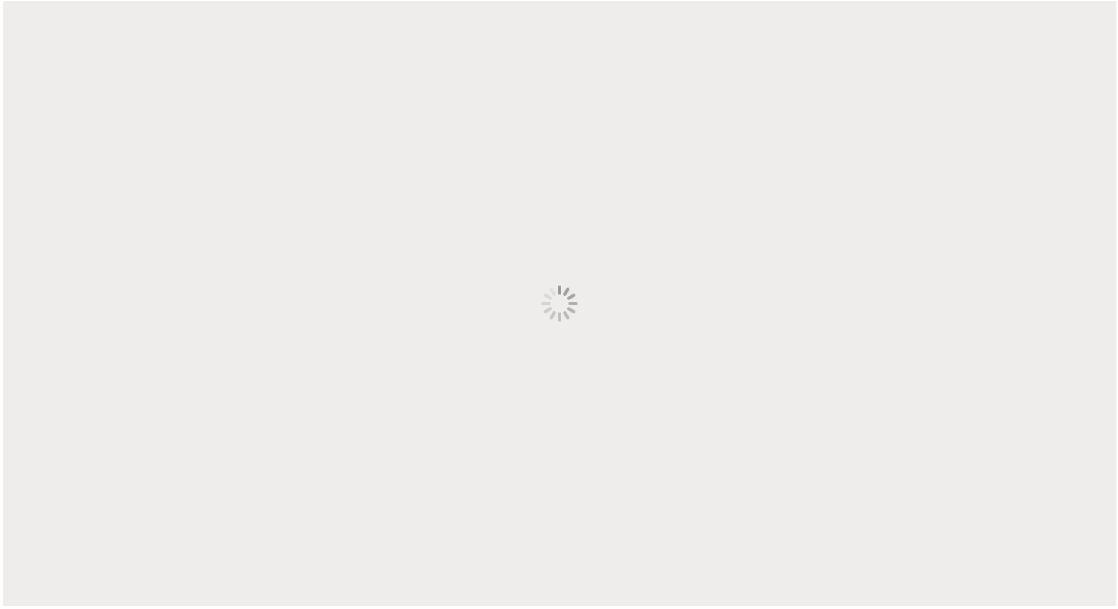
```
segment_image.segmentImage("path_to_image", output_image_name = "output_image_path
```

这是对图像进行实例分割的代码，它需要两个参数：

- path_to_image：模型所要预测图像的路径。
- output_image_name：保存分割结果的路径，将被保存在当前的工作目录中。

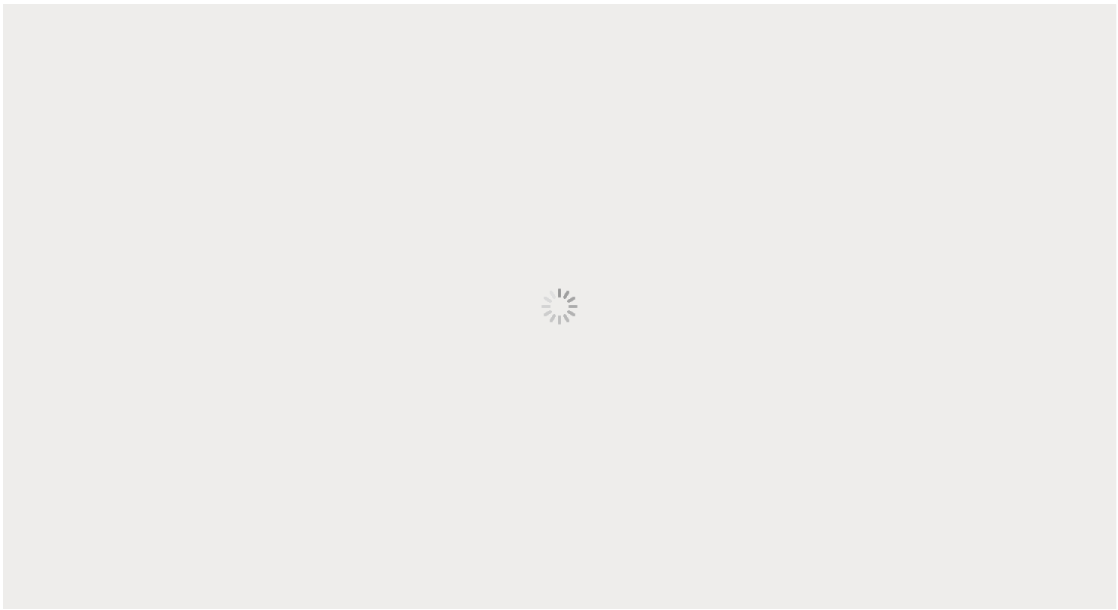
上图，实战第二弹！

图像文件命名为：sample2.jpg，如下图所示。



执行代码如下：

```
import pixellib
from pixellib.instance import instance_segmentation
segment_image = instance_segmentation()
segment_image.load_model("mask_rcnn_coco.h5")
segment_image.segmentImage("sample2.jpg", output_image_name = "image_new.jpg")
```

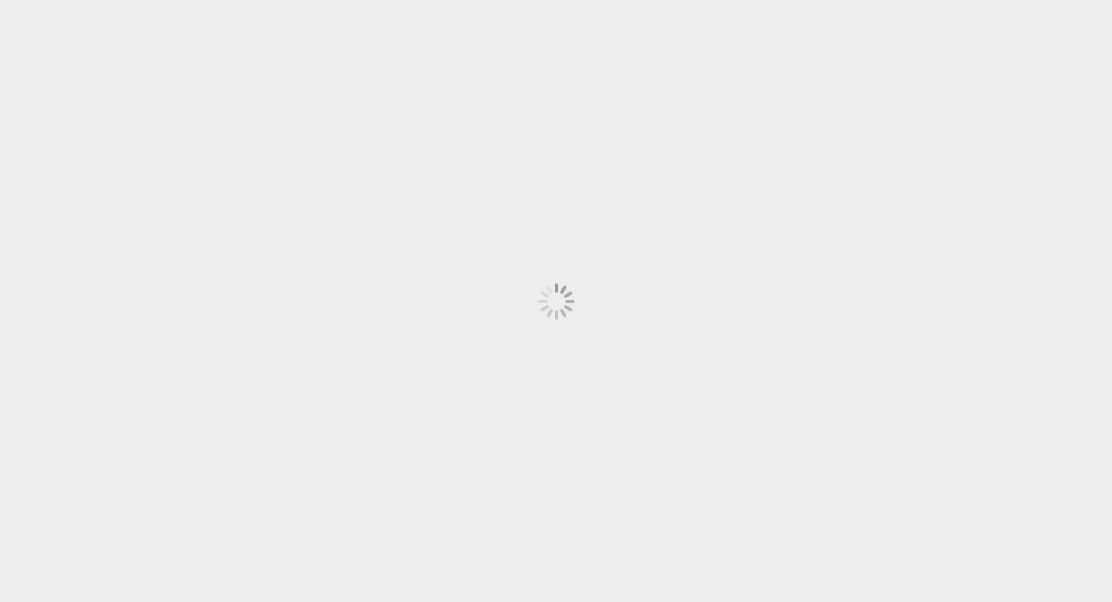


上图便是保存到目录的图片，现在可以看到语义分割和实例分割之间的明显区别——在实例分割中，同一类别的所有对象，都被赋予了不同的colormap。

若是想用边界框(bounding box)来实现分割，可以对代码稍作修改：

```
segment_image.segmentImage(“sample2.jpg”, output_image_name = “image_new.jpg”, sho
```

这样，就可以得到一个包含分割蒙版和边界框的保存图像。



同样的，也可以通过代码查询实例分割的推理时间：

```
import pixellib
from pixellib.instance import instance_segmentation
import time
segment_image = instance_segmentation()
segment_image.load_model(“mask_rcnn_coco.h5”)
start = time.time()
segment_image.segmentImage(“former.jpg”, output_image_name= “image_new.jpg”)
end = time.time()
print(f”Inference Time: {end-start:.2f}seconds”)
```

输出结果如下：

```
Inference Time: 12.55 seconds
```

可以看到，在图像上执行实例分割，需要12.55秒的时间。

最后，奉上项目、模型下载地址，快去试试吧~

传送门

PixelLib项目地址：https://github.com/ayoolaolafenwa/PixelLib

xception模型下载地址：
https://github.com/bonlime/keras-deeplab-v3-plus/releases/download/1.1/deeplabv3_xception_tf_dim_ordering_tf_kernels.h5

Mask RCNN模型下载地址：
https://github.com/matterport/Mask_RCNN/releases/download/v2.0/mask_rcnn_coco.h5

下载1：Pytoch常用函数手册

在「**OpenCV与AI深度学习**」公众号后台回复：**Pytorch常用函数手册**，即可下载全网第一份Pytorch常用函数手册，涵盖**Tensors介绍、基础函数介绍、数据处理函数、优化函数、CUDA编程、多线程处理**等十四章章内容。

下载2：145个OpenCV实例应用代码

在「**OpenCV與AI深度學習**」公眾號後台回復：**OpenCV145**，生成下載145個OpenCV實例應用代碼（**Python和C++雙語言實現**）。

-結束-

發現，麻煩給個贊，在看

收錄於故事#圖像處理 21

< 上一篇
計算機視覺中的傳統特徵提取方法總結
(SIFTHOG、SURF、ORB、LBP、HAAR等)

下一篇 >
實戰| 計算器/數碼管OCR數字識別(附源碼)

喜歡這個內容的人還喜歡

通過RAPID代碼記錄Tunemaster數據

ABB機器人實戰技巧