如何用TensorFlow和TF-Slim实现图像分类与分割

TensorFlow (/lib/tag/TensorFlow) 2016-11-29 15:14:25 发布

您的评价: 0.0

收藏 0收藏

本文将介绍如何用近日发布的TF-Slim工具包和预训练的模型来完成图像分类和图像分割。

引言

笔者将和大家分享一个结合了TensorFlow和 最近发布 的slim库的小应用，来实现图像分类、图像标注以及图像分割的任务，围绕着slim展开，包括其理论知识和应用场景。

之前自己尝试过许多其它的库，比如Caffe、Matconvnet、Theano和Torch等。它们各有优劣，而我想要一个可靠灵活的、自带预训练模型的python库。最近，新推出了一款名叫slim的库，slim自带了许多预训练的模型，比如ResNet、VGG、Inception-ResNet-v2（ ILSVRC的新赢家 ）等等。这个库和模型都是Google支持开发的。Google的Tensorflow是一个偏底层的库，实际使用时开发人员需要编写大量的代码，阅读他人的代码也很费劲，因此大家早就需要这样一个简洁的库。而slim非常干净，用预训练的模型对Tensorflow做了轻量级的封装。

下文中会用到Tensorflow和卷积神经网络的知识。Tensorflow的网站上有两者的完美教程，不了解的读者可以前去 阅读 。

我是用jupyter notebook完成的写作。因此，每一段代码之后都打印了运行结果。读者们也可以 下载 完整的notebook。本文中有一部分内容借鉴了 此文

安装

在运行代码之前，首先需要安装Tensorflow。我用的是0.11版本。你可以从github的tensorflow/models代码库克隆代码。

```
git clone https://github.com/tensorflow/models
```

我还会用到scikit-image和numpy等依赖，把它们都先装上。在这里我推荐先下载并安装Anaconda，然后通过conda install命令安装其它的python库。

首先，我们指定tensorflow使用第一块GPU。否则tensorflow默认会占用所有可用的内存资源。其次，添加克隆下来的代码库路径，这样python执行的时候就能找到需要的代码。

```
import sys
import os

os.environ["CUDA_VISIBLE_DEVICES"] = '0'
sys.path.append("/home/dpakhom1/workspace/models/slim")
```

接着，下载VGG-16模型，我们将用它来对图像做分类和分割。也可以选用其它占用内存少的网络模型（比如，AlexNet）。

```
from datasets import dataset_utils
import tensorflow as tf

url = "http://download.tensorflow.org/models/vgg_16_2016_08_28.tar.
gz"

# 指定保存路径
checkpoints_dir = '/home/dpakhom1/checkpoints'

if not tf.gfile.Exists(checkpoints_dir):
    tf.gfile.MakeDirs(checkpoints_dir)

dataset_utils.download_and_uncompress_tarball(url, checkpoints_dir)

>> Downloading vgg_16_2016_08_28.tar.gz 100.0%
    Successfully downloaded vgg_16_2016_08_28.tar.gz 513324920 byte
s.
```

图像分类

我们刚刚下载的模型可以将图像分成 1000类 。类别的覆盖度非常广。在本文中，我们就用这个预训练的模型来给图片分类、标注和分割，映射到这1000个类别。

下面是一个图像分类的例子。图像首先要做预处理，经过缩放和裁剪，输入的图像尺寸与训练集的图片尺寸相同。

```
%matplotlib inline

from matplotlib import pyplot as plt

import numpy as np
import os
import tensorflow as tf
import urllib2

from datasets import imagenet
from nets import vgg
from preprocessing import vgg_preprocessing

checkpoints_dir = '/home/dpakhom1/checkpoints'

slim = tf.contrib.slim

# 网络模型的输入图像有默认的尺寸
# 因此，我们需要先调整输入图片的尺寸
image_size = vgg.vgg_16.default_image_size

with tf.Graph().as_default():

url = ("https://upload.wikimedia.org/wikipedia/commons/d/d9/"
       "First_Student_IC_school_bus_202076.jpg")

# 连接网址，下载图片
image_string = urllib2.urlopen(url).read()

# 将图片解码成jpeg格式
image = tf.image.decode_jpeg(image_string, channels=3)

# 对图片做缩放操作，保持长宽比例不变，裁剪得到图片中央的区域
# 裁剪后的图片大小等于网络模型的默认尺寸
processed_image = vgg_preprocessing.preprocess_image(image,
                                                     image_size,
                                                     image_size,
                                                     is_training=Fa
lse)

# 可以批量导入图像
# 第一个维度指定每批图片的张数
# 我们每次只导入一张图片
```

```
processed_images  = tf.expand_dims(processed_image, 0)

# 创建模型，使用默认的arg scope参数
# arg_scope是slim library的一个常用参数
# 可以设置它指定网络层的参数，比如stride, padding 等等。
with slim.arg_scope(vgg.vgg_arg_scope()):
    logits, _ = vgg.vgg_16(processed_images,
                           num_classes=1000,
                           is_training=False)

# 我们在输出层使用softmax函数，使输出项是概率值
probabilities = tf.nn.softmax(logits)

# 创建一个函数，从checkpoint读入网络权值
init_fn = slim.assign_from_checkpoint_fn(
    os.path.join(checkpoints_dir, 'vgg_16.ckpt'),
    slim.get_model_variables('vgg_16'))

with tf.Session() as sess:

    # 加载权值
    init_fn(sess)

    # 图片经过缩放和裁剪，最终以numpy矩阵的格式传入网络模型
    np_image, network_input, probabilities = sess.run([image,
                                                       processed_im
age,
                                                       probabilitie
s])
    probabilities = probabilities[0, 0:]
    sorted_inds = [i[0] for i in sorted(enumerate(-probabilities),
                                        key=lambda x:x[1])]

# 显示下载的图片
plt.figure()
plt.imshow(np_image.astype(np.uint8))
plt.suptitle("Downloaded image", fontsize=14, fontweight='bold')
plt.axis('off')
plt.show()

# 显示最终传入网络模型的图片
# 图像的像素值做了[-1, 1]的归一化
# to show the image.
plt.imshow( network_input / (network_input.max() - network_input.mi
```

```
n()) )
plt.suptitle("Resized, Cropped and Mean-Centered input to network",
             fontsize=14, fontweight='bold')
plt.axis('off')
plt.show()

names = imagenet.create_readable_names_for_imagenet_labels()
for i in range(5):
    index = sorted_inds[i]
    # 打印top5的预测类别和相应的概率值。
    print('Probability %0.2f => [%s]' % (probabilities[index], name
s[index+1]))

res = slim.get_model_variables()
```

Downloaded image

Resized, Cropped and Mean-Centered input to the network

```
Probability 1.00 => [school bus]
Probability 0.00 => [minibus]
Probability 0.00 => [passenger car, coach, carriage]
Probability 0.00 => [trolleybus, trolley coach, trackless trolley]
Probability 0.00 => [cab, hack, taxi, taxicab]
```

图片标注和分割

从上面的例子中可以看到，网络模型只处理了原始图像中的一部分区域。这种方式只适用于单一预测结果的场景。

某些场景下，我们希望从图片中获得更多的信息。举个例子，我们想知道图片中出现的所有物体。网络模型就告诉我们图片中有一辆校车，还有几辆小汽车和几幢建筑物。这些信息可以协助我们搭建一个图片搜索引擎。以上就是一个图片标注的简单应用。

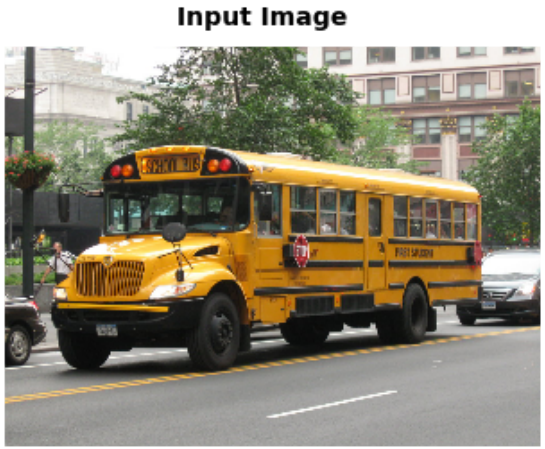
但是，如果我们也想得到物体的空间位置该怎么办。网络能告诉我们它在图片的中央看到一辆校车，在右上角看到几幢建筑物？这样，我们就可以创建一个更具体的搜索查询词：“我想要找到中间有一辆校车，左上角有几只花盆的所有符合要求的图片”。

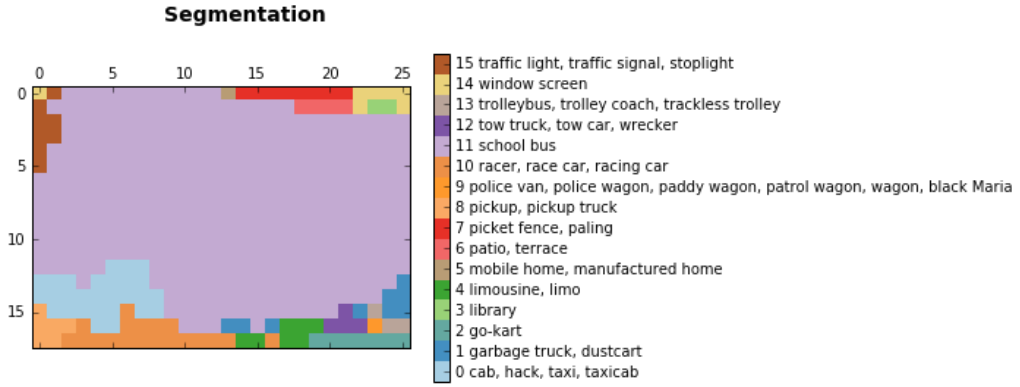
某些情况下，我们需要对图像的每个像素进行分类，也被称作是图像的分割。想象一下，假如有一个巨大的图片数据集，需要给人脸打上马赛克，这样我们就不必得到所有人的许可之后才能发布这些照片。例如，谷歌街景都对行人的脸做了模糊化处理。当然，我们只需要对图片中的人脸进行模糊处理，而不是所有的内容。图片分割可以帮助我们实现类似的需求。我们可以分割得到属于人脸的那部分像素，并只对它们进行模糊处理。

下面将介绍一个简单的图片分割例子。我们可以使用现有的卷积神经网络，通过完全卷积的方式进行分割。若想要输出的分割结果与输入图像尺寸保持一致，可以增加一个去卷积层。

```
from preprocessing import vgg_preprocessing

# 加载像素均值及相关函数
from preprocessing.vgg_preprocessing import (_mean_image_subtractio
n,
                                             _R_MEAN, _G_MEAN, _B_ME
AN)

# 展现分割结果的函数，以不同的颜色区分各个类别
def discrete_matshow(data, labels_names=[], title=""):
    #获取离散化的色彩表
    cmap = plt.get_cmap('Paired', np.max(data)-np.min(data)+1)
    mat = plt.matshow(data,
                      cmap=cmap,
                      vmin = np.min(data)-.5,
                      vmax = np.max(data)+.5)
    #在色彩表的整数刻度做记号
    cax = plt.colorbar(mat,

ticks=np.arange(np.min(data),np.max(data)+1))

    # 添加类别的名称
    if labels_names:
        cax.ax.set_yticklabels(labels_names)

    if title:
        plt.suptitle(title, fontsize=14, fontweight='bold')


with tf.Graph().as_default():

    url = ("https://upload.wikimedia.org/wikipedia/commons/d/d9/"
           "First_Student_IC_school_bus_202076.jpg")

    image_string = urllib2.urlopen(url).read()
    image = tf.image.decode_jpeg(image_string, channels=3)

    # 减去均值之前，将像素值转为32位浮点
    image_float = tf.to_float(image, name='ToFloat')

    # 每个像素减去像素的均值
    processed_image = _mean_image_subtraction(image_float,
                                              [_R_MEAN, _G_MEAN, _B
```

```
_MEAN])

    input_image = tf.expand_dims(processed_image, 0)

    with slim.arg_scope(vgg.vgg_arg_scope()):

        # spatial_squeeze选项指定是否启用全卷积模式
        logits, _ = vgg.vgg_16(input_image,
                               num_classes=1000,
                               is_training=False,
                               spatial_squeeze=False)

    # 得到每个像素点在所有1000个类别下的概率值，挑选出每个像素概率最大的类别
    # 严格说来，这并不是概率值，因为我们没有调用softmax函数
    # 但效果等同于softmax输出值最大的类别
    pred = tf.argmax(logits, dimension=3)

    init_fn = slim.assign_from_checkpoint_fn(
        os.path.join(checkpoints_dir, 'vgg_16.ckpt'),
        slim.get_model_variables('vgg_16'))

    with tf.Session() as sess:
        init_fn(sess)
        segmentation, np_image = sess.run([pred, image])

# 去除空的维度
segmentation = np.squeeze(segmentation)

unique_classes, relabeled_image = np.unique(segmentation,
                                            return_inverse=True)

segmentation_size = segmentation.shape

relabeled_image = relabeled_image.reshape(segmentation_size)

labels_names = []

for index, current_class_number in enumerate(unique_classes):

    labels_names.append(str(index) + ' ' + names[current_class_numb
er+1])

discrete_matshow(data=relabeled_image, labels_names=labels_names, t
itle="Segmentation")
```





我们得到的结果显示网络模型确实可以从图片中找到校车，以及左上角显示不太清晰的交通标志。而且，模型可以找到左上角建筑物的窗户，甚至猜测说这是一个图书馆（我们无法判断是否属实）。它做出了一些不那么正确的预测。这些通常是由于网络在预测的时候只能看到当前像素周围的一部分图像。网络模型表现出来的这种特性被称为感受视野。在本文中，我们使用的网络模型的感受视野是404像素。所以，当网络只能观察到校车的一部分图片时，与出租车和皮卡车混淆了。

正如我们在上面所看到的，我们得到了图片的一个简单分割结果。它不算很精确，因为最初训练网络是用来进实现分类任务，而不是图像分割。如果想得到更好的结果，我们还是需要重新训练一个模型。不管怎么说，我们得到的结果是可以用作图像标注的。

使用卷积神经网络进行图像分割，可以被看作是对输入图像的不同部分进行分类。我们将网络聚焦于某个像素点，进行预测判断，并输出该像素的类别标签。这样，我们给分类和分割的结果增加了空间信息。

小结

本文介绍了用slim库实现图像的分类和分割，并且简要阐述了技术原理。

自带预训练模型的slim库是一款强大而灵活的工具，可以配合tensorflow使用。由于最近刚刚发布，文档不是很完善，有时候甚至要阅读代码来帮助理解。Google正在加快进度完善后续的工作。

来自：http://geek.csdn.net/news/detail/126133

扩展阅读

TensorFow的基本使用 (/lib/view/open1451607994167.html)
有趣的机器学习概念纵览：从多元拟合，神经网络到深度学习，给每个感兴趣的人 (/lib/view/open1466248455067.html)
TensorFlow实战之Scikit Flow系列指导：Part 1 (/lib/view/open1448458877173.html)
Google深度学习笔记 TensorFlow实现与优化深度神经网络 (/lib/view/open1464179849027.html)
【机器学习】AlexNet 的tensorflow 实现 (/lib/view/open1455240920761.html)

为您推荐

有趣的机器学习概念纵览：从多元拟合，神经网络到深度学习，给每个感兴趣的人 (/lib/view/open1466248455067.html)
Android应用程序开发以及背后的设计思想深度剖析 (/lib/view/open1466070376316.html)
【机器学习】Tensorflow学习笔记 (/lib/view/open1455240879745.html)
卷积神经网络（cnn） 手写数字识别 (/lib/view/open1420686078859.html)
图像的傅里叶变换 (/lib/view/open1446773801576.html)

更多

TensorFlow (https://www.baidu.com/s?wd=site:open-open.com TensorFlow)
机器学习 (https://www.baidu.com/s?wd=site:open-open.com 机器学习)

同类热门新闻

1. Google最新开源Inception-ResNet-v2，进一步提升图像分类水准 (http://www.open-open.com/news/view/5a9a6489)
2. 谷歌发布TensorFlow 1.0，推出新的机器学习工具 (http://www.open-open.com/news/view/30b93faf)
3. 2016AI巨头开源IP盘点 50个最常用的深度学习库 (http://www.open-open.com/news/view/259931e3)
4. TensorFlow v0.10.0 RC0发布,一个表达机器学习算法的接口 (http://www.open-open.com/news/view/339df4e7)
5. 如何评价Tensorflow和其它深度学习系统 (http://www.open-open.com/news/view/1069a70)
6. 谷歌发布tf.Transform：一个TensorFlow数据预处理库 (http://www.open-open.com/news/view/2bbc688d)

同类热门经验

1. 机器学习之开源库大总结 (/lib/view/open1364432241437.html)
2. 机器学习(Machine Learning)&深度学习(Deep Learning)资料 (/lib/view/open1428112201271.html)
3. Caffe 深度学习框架上手教程 (/lib/view/open1421995285109.html)
4. 深度学习框架Keras简介 (/lib/view/open1430982565991.html)
5. 斯坦福大学怎样讲“情感分析” (/lib/view/open1421114964515.html)
6. 李航博士的《浅谈我对机器学习的理解》 机器学习与自然语言处理 (/lib/view/open1421287389750.html)

阅读目录