读书笔记 6 在 python 界面上用训练好的 Imagenet 模型去分类图形

2014.7.24 薛开宇

本篇是对前面第 5 个读书笔记的一个补充。讲一下在 python 界面上分类的一些细节。

注意，本笔记的代码由原文复制过来，再复制过去 ipython 可能会有些格式上的错误，建议使用原文（本文最后）的代码进行复制使用。

1.1 开始

在终端输入：

ipython notebook

启动编辑平台。

Caffe 提供一个普遍的 python 接口用来接 caffe 的模型。该网络在 python/caffe/pycaffe.py 中。

我们可以用 python 和 matlab 进行分类，然而，python 拥有更多功能，因此，我们在这里使用它，对于 matlab，可以参考 matlab/caffe/matcaffe_demo.m.

然后，准备工作和读书笔记 5 一样，这里不描述了。

首先，以下一段指令：

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

caffe_root = '/home/xuekaiyu/caffe-master/'# this file is expected to be in {caffe_root}/examples
import sys
sys.path.insert(0, caffe_root + 'python')
import caffe
# Set the right path to your model definition file, pretrained model weights,
#设置好模型路径，和想分类的图像
# and the image you would like to classify.
MODEL_FILE = caffe_root+'examples/imagenet/imagenet_deploy.prototxt'
PRETRAINED = caffe_root+'examples/imagenet/caffe_reference_imagenet_model'
IMAGE_FILE = caffe_root+'examples/images/bird11.jpeg'
```

1.2 加载网络与输入图片

加载一个网络很简单，caffe.Classifier 已经帮你设置好一切，注意输入预处理的参数配置，减去均值的文件的设置，输入的 RGB 频道的交换（ImageNet model's 是 BGR），还有输入时候乘以一定的特征比例以达到从【0，1】到【256】的目的。

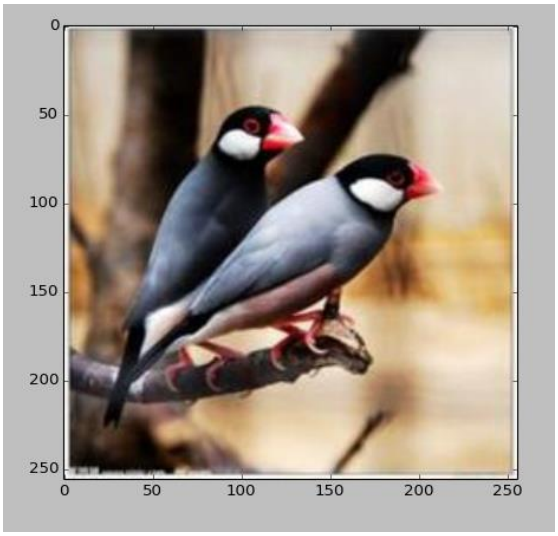
```
net = caffe.Classifier(MODEL_FILE, PRETRAINED,
                       mean_file=caffe_root+'python/caffe/imagenet/ilsvrc_2012_mean.npy',
                       channel_swap=(2,1,0),
                       input_scale=255)
```

因为是在测试阶段，而且我们先使用 CPU 计算，所以输入：

```
net.set_phase_test()
net.set_mode_cpu()
```

然后输入图片

```
input_image = caffe.io.load_image(IMAGE_FILE)
plt.imshow(input_image)
Plt.show()
```

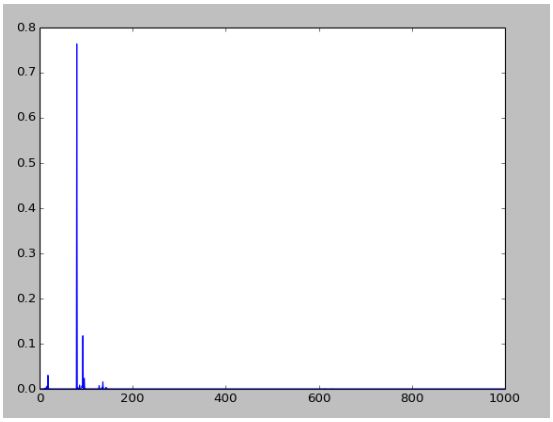


1.3 开始分类

默认是 10 种预测，裁剪照片的中心和边角，以及去掉他们的镜像版本，还有他们预测的均值。

（1）第一种分类政策：默认。

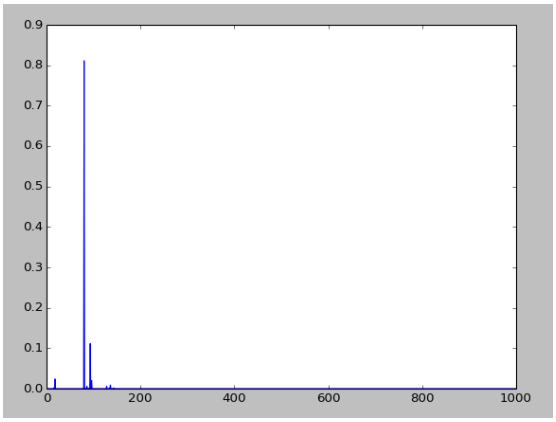
```
prediction = net.predict([input_image])  #预测可以输入任何大小的图像，caffe net 会自动调整。
print 'prediction shape:', prediction[0].shape
plt.plot(prediction[0])
```



（2）第二种分类政策：

现在我们只是用中心结果分类，关闭掉过采样，注意，对一个单一的输入，尽管我们检查模型定义 prototxt 我们可以看到网络的批次大小是 10，python 会自动处理和填充。

```
prediction=net.predict([input_image],oversample=False)
print 'prediction shape:', prediction[0].shape
plt.plot(prediction[0])
```



我们可以看到，预测是 1000 维，这样做更加分散。

我们的预训练模型使用同义词集 ID 类排序，在 ../data/ilsvrc12/synset_words.txt 可以看到对应，我们看指标，可以看到分数最大是【'n01795545 黑琴鸡”n01829413 犀鸟”n01582220 鹊 “n01843383 巨嘴鸟” n02017213 欧洲水鸡，紫水鸡’]，也算相当准确。

现在，我们可以看看执行分类的时间，这一结果是用 Intel I7 CPU，你可以看到一些性能上的差异。

1 loops, best of 3: 2.88 s per loop

这看起来有点慢，这是因为这是花费在接收获，python 的接口，并且运行 10 张照片。在性能上，可以选择编码在 C++和流水线上，实验的话当前速度是可以的。

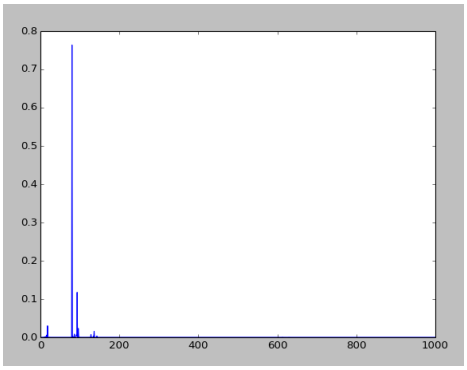
（3）第三种分类政策：

我们分类一张照片时，加入输入预处理。

```
# Resize the image to the standard (256, 256) and oversample net input sized crops.
#将照片的形状变成（256，256）同时进行过采样。
input_oversampled = caffe.io.oversample([caffe.io.resize_image(input_image, net.image_dims)], net.crop_dims)
# 'data' is the input blob name in the model definition, so we preprocess for that input.
# 数据输入的是 model 定义中 blob 的名字，所以我们预处理输入。
caffe_input = np.asarray([net.preprocess('data', in_) for in_ in input_oversampled])
# forward() takes keyword args for the input blobs with preprocessed input arrays.
# forward()为经过预处理输入的 blobs 获取关键字参数
%timeit net.forward(data=caffe_input)
```

使用的时间为 1 loops, best of 3: 2.73 s per loop，明显少了些。

让我们看看如果对我们的相同结果使用 GPU，从图可以看出，图没什么变化，那么时间来说：




```
# Full pipeline timing.
%timeit net.predict([input_image])
```

使用的时间，1 loops, best of 3: 289 ms per loop，少了非常多。之前是 S，现在是 ms 级别

```
# Forward pass timing.
%timeit net.forward(data=caffe_input)
```

使用的时间，10 loops, best of 3: 121 ms per loop，进一步减少。

事实上，使用 GPU 代码非常快，你可以看到几乎快了 10 倍，和数据加载，转换，接口都比本身的 ConvNet 计算的时间长。

若想发挥 GPU 的力量，你可以：

使用更大的批次，并且减少 Python 的调用和数据传输的开销。

管道数据加载操作，如使用 subprocess 模块来管理。

在 C++里编码。

1.4 后话

所以这是 python，我们希望接口非常简单。Python 的 boost 库和它源代码可以在

python/caffe 中发现。如果你想自己制作，从哪里开始，但是希望能给 caffe 团队知道你做出了改进并且提交。

资料来源：

http://nbviewer.ipython.org/github/BVLC/caffe/blob/master/examples/imagenet_classification.ipynb