# [利用训练好的Caffe网络得到输入图像的分类](http://blog.csdn.net/deeplearninglc007/article/details/41283985)

本文档描述是：如果已经训练好了一个Caffe网络，如何利用这个网络进行图像的分类。

以下以mnist网络为例。

mnist是用来分类手写数字0-9的。当用户写好一个数字后，图像进入mnist网络，然后网络计算出每个数的概率，认为读到的是概率最大的数。

**安装python**

        使用网络需要用到python，因此请安装好python，并将路径设置好，详见[http://caffe.berkeleyvision.org/installation.html](http://caffe.berkeleyvision.org/installation.html" \t "_blank)Python and/or Matlab wrapper(optional)章节

         找到$CAFFE\_ROOT\python\classify.py，将其复制到另外一个地方，例如$CAFFE\_ROOT\examples\mnist，命名为classifytest.py

         打开这个文件。该文件import了Caffe相关的文件，然后读入arg中的参数，如input\_file、output\_file、model\_def、pretrained\_model等等，有些参数看起来与我们要用的无关，例如channel\_swap (mnist输入的是黑白图像，只有单个通道)、input\_scale(默认为1)等等，但是这些参数的代码段**千万不能删除，删除可能导致之后调用python时出错。**如果你觉得在终端中输入这些参数太麻烦，可以加个【dafault=你希望的默认值】

**修改参数**

     将参数中的"--model\_def"的值改为网络结构的prototxt，例如我是这样改的：

**[plain]** [view plaincopy](http://blog.csdn.net/deeplearninglc007/article/details/41283985)

1. parser.add\_argument(
2. "--model\_def",
3. default=os.path.join(pycaffe\_dir,
4. "../../examples/mnist/lenet.prototxt"),
5. help="Model definition file."

      将参数pretrained\_model改为之前训练好的参数文件，例如我是这样改的：

**[plain]** [view plaincopy](http://blog.csdn.net/deeplearninglc007/article/details/41283985)

1. parser.add\_argument(
2. "--pretrained\_model",
3. default=os.path.join(pycaffe\_dir,
4. "../../examples/mnist/lenet\_iter\_10000.caffemodel"),
5. help="Trained model weights file."

     由于mnist要求输入的图像是灰度图像，因此需要将RGB图转换为灰度图，并更改channel\_swap：

在文件的开头加入

from skimage.color import rgb2gray

     在将图片读入inputs后将RGB图转换为灰度图

inputs = [rgb2gray(input) for input in inputs]

更改channel\_swap:

channel\_swap = [0]

.py文件更改就是这样了，如果你的模型不一样，需要采取不一样的更改，举一反三。

准备好一张图片作为输入图片，这张图片必须是28\*28大小的JPEG格式图片，内容是随意的，但是毕竟mnist用于分类数字，我们用windows的画图工具画好一个黑底白字的数字’0’http://img.blog.csdn.net/20141119164532796?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvZGVlcGxlYXJuaW5nbGMwMDc=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center，然后把这个图片传到$CAFFE\_ROOT\examples\mnist下，命名为testformnist0.jpg

**进入终端：**

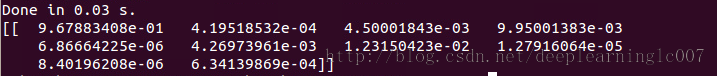
 cd $CAFFE\_ROOT

python examples/mnist/classifytest.pyexamples/mnist/testformnist0.jpg output

http://img.blog.csdn.net/20141119164654781?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvZGVlcGxlYXJuaW5nbGMwMDc=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

       第一个参数是python，表示使用python去执行第二个参数。第三个参数是输入文件的位置input\_file，第四个参数是输出文件的位置output\_file

结果如下：



其中Probability(input==’0’)=0.967，网络正确的分类了这个图片。

如果想要显示中间的某个结果，则可以通过修改网络结构定义文件来实现。例如，我想要看看在softmax\_loss层之前的结果是什么：

打开lenet\_prototxt

文件最后是

**[plain]** [view plaincopy](http://blog.csdn.net/deeplearninglc007/article/details/41283985)

1. layers {
2. name: "prob"
3. type: SOFTMAX
4. bottom: "ip2"
5. top: "prob"
6. }

      表示有个层，输入ip2(inner\_product2)，输出prob(每个数的概率)，层的类型是softmax，并且综合整个网络结果，这个层是最顶层，其输出将会作为整个网络的输出。

因此我们只需要删除这个层，那么以前的最顶层就被次顶层替代了，次顶层的结果就是softmax之前的运算结果。

删除之后再运行python:

http://img.blog.csdn.net/20141119164853453?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQvZGVlcGxlYXJuaW5nbGMwMDc=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast

完全看不懂了是不是？

确实是这样的，deep learning网络的参数和运行结果很不容易理解，某些时候确实需要网络的中间结果。