[caffe python visualization程序解析](http://blog.csdn.net/guoyilin/article/details/42873747)

分类： [深度学习](http://blog.csdn.net/guoyilin/article/category/2931141)2015-01-19 15:35 1498人阅读 [评论](http://blog.csdn.net/guoyilin/article/details/42873747#comments)(8) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/guoyilin/article/details/42873747#report)

本文主要对http://nbviewer.ipython.org/github/BVLC/caffe/blob/master/examples/filter\_visualization.ipynb进行代码解析。  
1. net.blobs.items() 存储了预测图片的网络中各层的feature map的数据。  
2. net.params.items()存储了训练结束后学习好的网络参数。  
3. vis\_square 函数视觉化data，主要是进行数据归一化，data转换为plt可视化的square结构。  
4.   
7.

plt.imshow(net.deprocess('data', net.blobs['data'].data[4]))

这里的4是第4个crop，图片会被crop成10个227\*227.

5. net.params['conv1'][0].data， 这是表示conv1层的w参数

  net.params['conv2'][1].data, 这是表示conv1层的b参数

6.

filters.transpose(0, 2, 3, 1)对filters 4维数组进行位置对换，主要是为了将rgb放在最后一维。

net.blobs['conv1'].data[4, :36] 表示conv1层学习的feature map， 显示第4个crop image的top 36个feature map。

8.

filters = net.params['conv2'][0].data

filters[:48].reshape(48\*\*2, 5, 5) 对conv2 层参数w进行显示， conv2 ：256 \* 48 \* 5 \* 5, 这里显示头48个filters， reshape是为了在显示的时候把48个5\*5的kernel放在一行显示，共48\*48的方格显示。

# [affe python批量抽取图像特征](http://blog.csdn.net/guoyilin/article/details/42886365)

分类： [机器学习](http://blog.csdn.net/guoyilin/article/category/2187399)2015-01-19 22:36 2833人阅读 [评论](http://blog.csdn.net/guoyilin/article/details/42886365#comments)(15) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/guoyilin/article/details/42886365#report)

[caffe](http://www.csdn.net/tag/caffe)[feature](http://www.csdn.net/tag/feature)[python](http://www.csdn.net/tag/python)

caffe在[1]讲到如何看一个图片的特征和分类结果，但是如何批量抽取特征呢？可以使用c++的版本，这里我们谈下如何用python批量抽取特征。

首先，我们要注意caffe filter\_visualization.ipynb的程序中deploy.prototxt中网络每一轮的图片batch是10， 这个数刚好和oversample=true的crop数量是一样的，也就是net一轮forward 刚好是一张图片的10个crop。

第一种，oversample = true的情况, 也就是每张图片会产生10张crop的图片: center, 4 corner, 和mirror

假如我们要抽取两张图片， 每张图片有10个crop

首先是修改deploy.prototxt: input\_dim : 20

然后：将imagelist 放入predict参数。

**[python]** [view plaincopy](http://blog.csdn.net/guoyilin/article/details/42886365)[在CODE上查看代码片](https://code.csdn.net/snippets/583934)

1. scores = net.predict([caffe.io.load\_image(caffe\_root + "building.jpg"), caffe.io.load\_image(caffe\_root + "thumb.jpg")])

最后用

**[python]** [view plaincopy](http://blog.csdn.net/guoyilin/article/details/42886365)[在CODE上查看代码片](https://code.csdn.net/snippets/583934)

1. net.blobs['fc7'].data[4]
2. net.blobs['fc7'].data[14]

**[python]** [view plaincopy](http://blog.csdn.net/guoyilin/article/details/42886365)[在CODE上查看代码片](https://code.csdn.net/snippets/583934)

1. **import** numpy as np
2. **import** scipy
3. caffe\_root = '/home/hduser/Project/caffe/'
4. **import** sys
5. sys.path.insert(0,caffe\_root + 'python/')
6. **import** caffe
8. net = caffe.Classifier(caffe\_root + 'models/bvlc\_reference\_caffenet/deploy.prototxt',
9. caffe\_root + 'models/bvlc\_reference\_caffenet/bvlc\_reference\_caffenet.caffemodel',image\_dims=(256, 256))
10. net.set\_phase\_test()
11. net.set\_mode\_cpu()
12. net.set\_mean('data', np.load(caffe\_root + 'python/caffe/imagenet/ilsvrc\_2012\_mean.npy'))
13. net.set\_raw\_scale('data', 255)
14. net.set\_channel\_swap('data', (2,1, 0))
16. #in fact, you can input a list of images.
17. scores = net.predict([caffe.io.load\_image(caffe\_root + "building.jpg"), caffe.io.load\_image(caffe\_root + "thumb.jpg")])
18. output = open("feature.txt", "w")
20. #the fc6 is the fc6 layer feature, data[4] means the five crop images, because each image will be crop to 10 sub-images.
21. feat = net.blobs['fc6'].data[4]
22. feat2 = net.blobs['fc6'].data[14]

这样不好的地方是需要修改deploy.prototxt， 另一种方法[2]：modify predict() in python/caffe/classifier.py to store them before the blobs in net are overwritten by the features of a subsequent batch. 该种方法我还没尝试，改天试下。

**[python]** [view plaincopy](http://blog.csdn.net/guoyilin/article/details/42886365)[在CODE上查看代码片](https://code.csdn.net/snippets/583934)

1. <span style="font-family: Arial, Helvetica, sans-serif; background-color: rgb(255, 255, 255);">第二种情况， 如果图片不需要crop成10张子图片的话，可以用oversample=False，如果设置image\_dims=(256, 256), 由于bvlc\_reference\_caffenet trained model 是227\*227的图片大小，所以python/caffe/classifier.py的代码会take center crop 227\*227 **from** 256\*256.</span>

input\_dim : 2

**[python]** [view plaincopy](http://blog.csdn.net/guoyilin/article/details/42886365)[在CODE上查看代码片](https://code.csdn.net/snippets/583934)

1. <span style="font-family: Arial, Helvetica, sans-serif; background-color: rgb(255, 255, 255);">第一张图片：net.blobs['fc7'].data[0],</span>

**[python]** [view plaincopy](http://blog.csdn.net/guoyilin/article/details/42886365)[在CODE上查看代码片](https://code.csdn.net/snippets/583934)

1. <span style="font-family: Arial, Helvetica, sans-serif; background-color: rgb(255, 255, 255);">第二张图片：net.blobs['fc7'].data[1]</span>

后记：还需要熟悉下caffe的python接口函数，可惜貌似看不到这方面的api，只有自己先琢磨琢磨。

参考文章：

1. http://nbviewer.ipython.org/github/BVLC/caffe/blob/master/examples/filter\_visualization.ipynb

2. https://groups.google.com/forum/#!searchin/caffe-users/python$20batch$20feature$20extraction/caffe-users/wIgLYMF54AI/iuDf3fZ0\_K0J

[caffe c++ 抽取图片特征](http://blog.csdn.net/guoyilin/article/details/42886483)

分类： [机器学习](http://blog.csdn.net/guoyilin/article/category/2187399)2015-01-19 22:47 2936人阅读 [评论](http://blog.csdn.net/guoyilin/article/details/42886483#comments)(13) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/guoyilin/article/details/42886483#report)

[feature](http://www.csdn.net/tag/feature)[caffe](http://www.csdn.net/tag/caffe)[c++](http://www.csdn.net/tag/c%2b%2b)

caffe c++批量抽取特征的方法在[1]，但是该方法使用中有几个疑问：

1. 如何转换levelDB 格式为libsvm格式。

2.  ./build/tools/extract\_features mini-batch 是代表什么意思，和imagenet\_val.prototxt中的batch\_size的关系是什么？

本文主要解决如上两个问题，具体extract\_features源代码还需要进一步分析。

第一个问题，

**[plain]** [view plaincopy](http://blog.csdn.net/guoyilin/article/details/42886483)[在CODE上查看代码片](https://code.csdn.net/snippets/583941)

1. ./build/tools/extract\_features models/bvlc\_reference\_caffenet/bvlc\_reference\_caffenet.caffemodel examples/\_temp/imagenet\_val.prototxt fc7 examples/\_temp/features 10

其中，10 是mini-batch, 假设imagenet\_val.prototxt的batches size是128， 那么程序将抽取128 \* 10个图片的特征。如果你有100张图片， 你可以设置mini-batch = 1, batches size = 100.  
如果你的image个数是1283， 那么如上数值的设置会是的leveldb多出3个无用的feature，这个需要注意， 我测试过好像是会重复之前的图片，具体需要研究源代码。  
  
  
第二个问题， 特征保存的格式为leveldb，如果需要用libsvm的格式访问特征，可以用python 进行转换， 程序如下， 这里感谢bean的程序[1]：

**[python]** [view plaincopy](http://blog.csdn.net/guoyilin/article/details/42886483)[在CODE上查看代码片](https://code.csdn.net/snippets/583941)

1. **import** numpy as np
2. **import** caffe
3. **import** sys
4. **from** caffe.proto **import** caffe\_pb2
6. #parse argument
7. dbName = sys.argv[1]
8. featureFile = sys.argv[2]
9. output = open(featureFile, 'w')

**[python]** [view plaincopy](http://blog.csdn.net/guoyilin/article/details/42886483)[在CODE上查看代码片](https://code.csdn.net/snippets/583941)

1. # open leveldb files
2. db = leveldb.LevelDB(dbName)
4. # get db iterator
5. it = db.RangeIter()
6. count = 0
7. **for** key,value **in** it:
8. # convert string to datum
10. datum = caffe\_pb2.Datum.FromString(db.Get(key))
12. # convert datum to numpy string
14. arr = caffe.io.datum\_to\_array(datum)[0]
16. i = 0
17. tmpS = ''
19. # convert to svm format
21. **for** i **in** range(0, len(arr)):
22. tmpS += str(i+1) + ':' + str(arr[i].tolist()[0]) + ' '
23. #print tmpS
24. output.write(tmpS.strip() + "\n")
25. count+=1
26. **print** count
27. output.close()

但是这个程序有个巨大的bug，db.RangeIter()返回的key 顺序是按照 字母 进行排序的，和levelDB的排序方式是不一样的。具体参见[3]:

The problem is most likely caused by re-ordering of training/test examples since the db.RangeIter()iterates over keys in alphabetical order while extract\_features creates keys from index values without leading zeros (unlike convert\_imageset). Hence, you get an order like 0, 1, 10, 100, ...

Parse the key value in python and put the extracted feature vector at that position.

在这里，我们也只能说fuck了。修改后代码如下 ：

**[python]** [view plaincopy](http://blog.csdn.net/guoyilin/article/details/42886483)[在CODE上查看代码片](https://code.csdn.net/snippets/583941)

1. # get db iterator
2. it = db.RangeIter()
3. features = {}
4. **for** key,value **in** it:
5. # convert string to datum
6. datum = caffe\_pb2.Datum.FromString(value)
7. # convert datum to numpy string
8. arr = caffe.io.datum\_to\_array(datum)[0]
9. features[int(key)] = arr
11. #write to file, since the key in it is sorted by alpha\_number default, while leveldb is sorted by number, we must sort the key again.
12. sort\_features = collections.OrderedDict(sorted(features.items()))
13. **for** k, arr  **in** sort\_features.iteritems():
14. **if**(k > imageCount - 1):
15. **break**
16. line = ""
17. **for** i **in** range(0, len(arr)):
18. line += str(i+1) + ':' + str(arr[i].tolist()[0]) + ' '
19. output.write(line.strip() + "\n")
20. output.close()

通过对比c++和python提取的feature， 大部分是一致的，但是还是有会细微的差别，可能是浮点数的精度问题吧。

参考文章：

1. http://caffe.berkeleyvision.org/gathered/examples/feature\_extraction.html

2. http://bean.logdown.com/posts/211192-caffe-use-caffe-to-extract-features-of-each-layer

3. https://github.com/BVLC/caffe/issues/1158

[caffe python 批量抽取图像特征---续篇](http://blog.csdn.net/guoyilin/article/details/43420711)

分类： [深度学习](http://blog.csdn.net/guoyilin/article/category/2931141)2015-02-02 23:08 796人阅读 [评论](http://blog.csdn.net/guoyilin/article/details/43420711#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/guoyilin/article/details/43420711#report)

[python](http://www.csdn.net/tag/python)[caffe](http://www.csdn.net/tag/caffe)

http://blog.csdn.net/guoyilin/article/details/42886365 上文是使用修改deploy.prototxt文件的input\_dim为我们想要的图片的张数，因为net.blobs每当forward执行完毕一次batch， 都会更新网络的blobs数据。本文介绍在不修改deploy.prototxt情况下，更方便抽取图片特征。

首先，修改python/caffe/classifier.py, 修改predict方法

**[python]** [view plaincopy](http://blog.csdn.net/guoyilin/article/details/43420711)[在CODE上查看代码片](https://code.csdn.net/snippets/596104)

1. out = self.forward\_all(blobs=['fc7'], \*\*{self.inputs[0]: caffe\_in})
2. predictions = out[self.outputs[0]].squeeze(axis=(2,3))
3. fc7 = out['fc7'].squeeze(axis=(2,3))
4. **return** fc7

重新编译下caffe的python:

cd build

make pycaffe

然后copy 编译好的包到源码路径下：

**[plain]** [view plaincopy](http://blog.csdn.net/guoyilin/article/details/43420711)[在CODE上查看代码片](https://code.csdn.net/snippets/596104)

1. cp build/python/libpycaffe.so python/caffe/\_caffe.so && \
2. mkdir -p python/caffe/proto && \
3. touch python/caffe/proto/\_\_init\_\_.py && \
4. protoc --proto\_path=src/caffe/proto --python\_out=python/caffe/proto src/caffe/proto/caffe.proto && \
5. protoc --proto\_path=src/caffe/proto --python\_out=python/caffe/proto src/caffe/proto/caffe\_pretty\_print.proto

主程序文件调用如下：

**[python]** [view plaincopy](http://blog.csdn.net/guoyilin/article/details/43420711)[在CODE上查看代码片](https://code.csdn.net/snippets/596104)

1. features  = net.predict(imageList, oversample=False)
2. **for** feature **in** features: