

[首页](#)[导航](#)[评测](#)[3D打印](#)[发现](#)[商城](#)[投稿](#)[帖子](#)[登录/注册](#)[3D打印](#)[人工智能AI/机器学习](#)[Facebook Caffe](#)[caffe 提取特征并可视化（已测试可执行）及在线可视化1](#)**东方熊1**

286

0

5 天前

作者

[发表帖子](#)

caffe 提取特征并可视化（已测试可执行）及在线可视化1 [\[复制链接\]](#)

参考主页：

caffe 可视化的资料可在百度云盘下载

链接: <http://pan.baidu.com/s/1jIRJ6mU>

提取密码：xehi

<http://cs.stanford.edu/people/karpathy/cnnembed/>[http://lijiancheng0614.github.io ...](http://lijiancheng0614.github.io...) 21 CAFFE Features/[http://nbviewer.ipynb.org/gith ...](http://nbviewer.ipynb.org/gith...) lassification.ipynb<http://www.cnblogs.com/platero/p/3967208.html>[http://caffe.berkeleyvision.org/ ...](http://caffe.berkeleyvision.org/...) ure extraction.html<http://caffecn.cn/?/question/21>

caffe程序是由c++语言写的，本身是不带数据可视化功能的。只能借助其它的库或接口，如opencv, python或matlab。使用python接口来进行可视化，因为python出了个比较强大的东西：ipython notebook, 最新版本改名叫jupyter notebook，它能将python代码搬到浏览器上去执行，以富文本方式显示，使得整个工作可以以笔记的形式展现、存储，对于交互编程、学习非常方便。

使用CAFFE（<http://caffe.berkeleyvision.org>）运行CNN网络，并提取出特征，将其存储成lmdb以供后续使用，亦可以对其

南极熊推荐

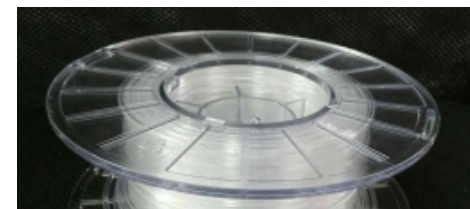


320万起售！履带式混凝土
1348人已关注

[参与](#)

材料名称	增强材料	拉伸强度 (MPa)	抗压强度 (MPa)	弹性模量 (GPa)
碳纤维	12-12	350	350	180
碳纤维	40x2+40x2	187	168	122
碳纤维	40x2+40x2	1128	918	100
碳纤维	87x8+87x8	1428	952	100
碳纤维	98x8+98x8	1068	1150	100
碳纤维	204x304	823	740	100
碳纤维	321x321	623	516	100

曝光：固相增材制造技术，
788人已关注

[参与](#)

可视化

化工巨头杜邦出手！携手

参与

595人已关注

使用已训练好的模型进行图像分类在 <http://nbviewer.ipython.org/github...lassification.ipynb> 中已经很详细地介绍了怎么使用已训练好的模型对测试图像进行分类了。由于CAFFE不断更新，这个页面的内容和代码也会更新。以下只记录当前能运行的主要步骤。下载CAFFE，并安装相应的dependencies，说白了就是配置好caffe。

- ```
01. 1. 下载CAFFE，并安装相应的dependencies，说白了就是配置好caffe运行环境。
02.
03. 2. 配置好 python ipython notebook,具体可参考网页:
 复制代码
```

<http://blog.csdn.net/jiandanjinxin/article/details/50409448>

3. 在caffe\_root下运行./scripts/download\_model\_binary.py models/bvlc\_reference\_caffenet获得预训练的CaffeNet。获取CaffeNet网络并储存到models/bvlc\_reference\_caffenet目录下。

- ```
01. cd caffe-root
02. python ./scripts/download_model_binary.py models/bvlc_reference_caffenet
    复制代码
```

- ```
01. 4. 在python文件夹下进入ipython模式（或python，但需要把部分代码注释掉）运行以下代码
 复制代码
```

- ```
01. cd ./python #（×）后面会提到
02. ipython notebook
    复制代码
```

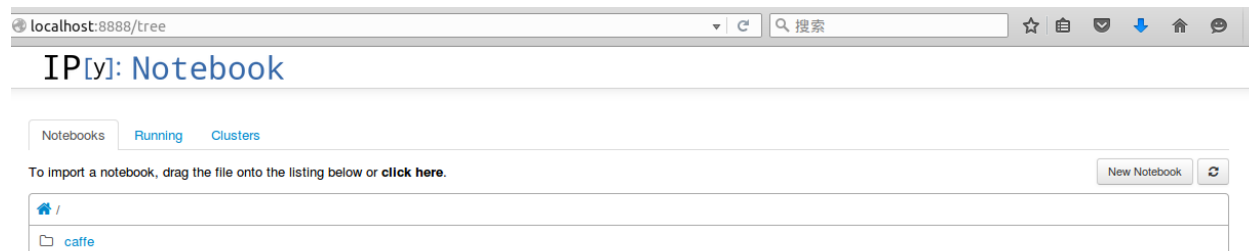
通知

社区首发

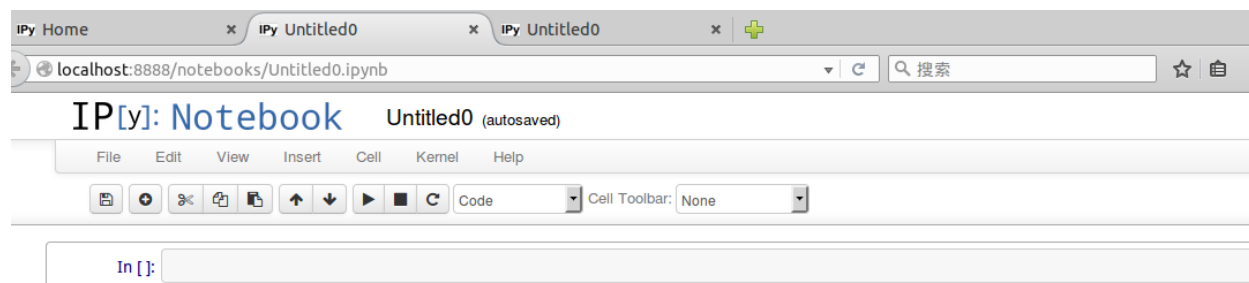
每周迭代

太空灰3D打印控制程序系统界面截图
金属零件3D打印技术在机械加工行业中..
了解3D建筑打印项目从哪几方面研究？..
金属3D打印激光束、电子束和等离子束..
太空灰3D建筑打印“太空灰”是什么灰
FDM打印机打印组合件的技巧
怎么样了解3D建筑打印项目？
解决3D打印机马达失步的方法
太空灰3D建筑打印机 第二代3D建筑打..
太空灰3D建筑打印 真正的3D打印建筑

在命令行输入 `ipython notebook`,会出现一下画面



01. 接着 点击 New Notebook ,就可以输入代码,按 shift+enter 执行
复制代码



python环境不能单独配置，必须要先编译好caffe，才能编译python环境。

安装jupyter

01. `sudo pip install jupyter`
复制代码

安装成功后，运行notebook

01. `jupyter notebook`
复制代码

01. 输入下面代码：

复制代码

```
01. import numpy as np
02. import matplotlib.pyplot as plt
03. %matplotlib inline
04.
05. # Make sure that caffe is on the python path:
06. caffe_root = '../'
07. # this file is expected to be in {caffe_root}/examples
08. #这里注意路径一定要设置正确,记得前后可能都有"/",路径的使用是
09. #{caffe_root}/examples,记得 caffe-root 中的 python 文件夹需要包括 caffe 文件夹。
10.
11. #caffe_root = '/home/bids/caffer-root/' #为何设置为具体路径反而不能运行呢
12.
13. import sys
14. sys.path.insert(0, caffe_root + 'python')
15. import caffe #把 ipython 的路径改到指定的地方(这里是说刚开始在终端输入ipython notebook命令时,一定要确保是在包含
caffe的python文件夹,这就是上面代码(×)),以便可以调入 caffe 模块,如果不改路径,import 这个指令只会在当前目录查找,是找
不到 caffe 的。
16.
17. plt.rcParams['figure.figsize'] = (10, 10)
18. plt.rcParams['image.interpolation'] = 'nearest'
19. plt.rcParams['image.cmap'] = 'gray'
20. #显示的图表大小为 10,图形的插值是以最近为原则,图像颜色是灰色
21.
22. import os
23. if not os.path.isfile(caffe_root + 'models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel'):
24.     print("Downloading pre-trained CaffeNet model...")
25.     !../scripts/download_model_binary.py ../models/bvlc_reference_caffenet
26.
27. #设置网络为测试阶段,并加载网络模型prototxt和数据平均值mean_npy
28.
29. caffe.set_mode_cpu()# 采用CPU运算
30. net = caffe.Net(caffe_root + 'models/bvlc_reference_caffenet/deploy.prototxt',caffe_root +
'models/bvlc_reference_caffenet/bvlc_reference_caffenet.caffemodel',caffe.TEST)
31.
```

```
32. # input preprocessing: 'data' is the name of the input blob == net.inputs[0]
33. transformer = caffe.io.Transformer({'data': net.blobs['data'].data.shape})
34. transformer.set_transpose('data', (2,0,1))
35. transformer.set_mean('data', np.load(caffe_root +
    'python/caffe/imagenet/ilsvrc_2012_mean.npy').mean(1).mean(1))
36. # mean pixel, ImageNet的均值
37. transformer.set_raw_scale('data', 255)
38. # the reference model operates on images in [0,255] range instead of [0,1]。参考模型运行在【0,255】的灰度,而
    不是【0,1】
39.
40. transformer.set_channel_swap('data', (2,1,0))
41.
42. # the reference model has channels in BGR order instead of RGB, 因为参考模型本来频道是 BGR, 所以要将RGB转换
43.
44. # set net to batch size of 50
45. net.blobs['data'].reshape(50,3,227,227)
46.
47. #加载测试图片,并预测分类结果。
48.
49. net.blobs['data'].data[...] = transformer.preprocess('data', caffe.io.load_image(caffe_root +
    'examples/images/cat.jpg'))
50. out = net.forward()
51. print("Predicted class is #{0}.".format(out['prob'][0].argmax()))
52.
53. plt.imshow(transformer.deprocess('data', net.blobs['data'].data[0]))
54.
55. # load labels, 加载标签, 并输出top_k
56. imagenet_labels_filename = caffe_root + 'data/ilsvrc12/synset_words.txt'
57. try:
58.     labels = np.loadtxt(imagenet_labels_filename, str, delimiter='\t')
59. except:
60.     !../data/ilsvrc12/get_ilsvrc_aux.sh
61.     labels = np.loadtxt(imagenet_labels_filename, str, delimiter='\t')
62. # sort top k predictions from softmax output
63.
64. top_k = net.blobs['prob'].data[0].flatten().argsort()[-1:-6:-1]
65. print labels[top_k]
66.
```

```
67.
68. # CPU 与 GPU 比较运算时间
69. # CPU mode
70.
71. net.forward() # call once for allocation
72. %timeit net.forward()
73.
74. # GPU mode
75. caffe.set_device(0)
76. caffe.set_mode_gpu()
77. net.forward() # call once for allocation
78. %timeit net.forward()
79.
80. *****提取特征并可视化*****
81.
82. #网络的特征存储在net.blobs，参数和bias存储在net.params，以下代码输出每一层的名称和大小。这里亦可手动把它们存储下来。
83.
84. [(k, v.data.shape) for k, v in net.blobs.items()]
85.
86. #显示出各层的参数和形状，第一个是批次，第二个 feature map 数目，第三和第四是每个神经元中图片的长和宽，可以看出，输入是
    227*227 的图片，三个频道，卷积是 32 个卷积核卷三个频道，因此有 96 个 feature map
87.
88. [(k, v[0].data.shape) for k, v in net.params.items()]
89. #输出：一些网络的参数
90.
91. ***可视化的辅助函数**
92. # take an array of shape (n, height, width) or (n, height, width, channels)用一个格式是(数量,高,宽)或(数量,
    高,宽,频道)的阵列
93. # and visualize each (height, width) thing in a grid of size approx. sqrt(n) by sqrt(n)每个可视化的都是在一个
    由一个个网格组成
94.
95. def vis_square(data, padsizes=1, padval=0):
96.     data -= data.min()
97.     data /= data.max()
98.
99.     # force the number of filters to be square
100.    n = int(np.ceil(np.sqrt(data.shape[0])))
101.    padding = ((0, n ** 2 - data.shape[0]), (0, padsizes), (0, padsizes)) + ((0, 0),) * (data.ndim - 3)
```

```
102.     data = np.pad(data, padding, mode='constant', constant_values=(padval, padval))
103.
104.     # tile the filters into an image
105.     data = data.reshape((n, n) + data.shape[1:]).transpose((0, 2, 1, 3) + tuple(range(4, data.ndim +
106.     1)))
107.     data = data.reshape((n * data.shape[1], n * data.shape[3]) + data.shape[4:])
108.
109.     plt.imshow(data)
110.
111.     #根据每一层的名称,选择需要可视化的层,可以可视化filter (参数)和output (特征)
112.     # the parameters are a list of [weights, biases],各层的特征,第一个卷积层,共96个过滤器
113.     filters = net.params['conv1'][0].data
114.     vis_square(filters.transpose(0, 2, 3, 1))
115.     #使用 ipt.show()观看图像
116.
117.     #过滤后的输出,96 张 featuremap
118.     feat = net.blobs['conv1'].data[4, :96]
119.     vis_square(feat, padval=1)
120.     #使用 ipt.show()观看图像:
121.
122.     feat = net.blobs['conv1'].data[0, :36]
123.     vis_square(feat, padval=1)
124.
125.
126.     #第二个卷积层:有 128 个滤波器,每个尺寸为 5X5X48。我们只显示前面 48 个滤波器,每一个滤波器为一行。输入:
127.     filters = net.params['conv2'][0].data
128.     vis_square(filters[:48].reshape(48**2, 5, 5))
129.     #使用 ipt.show()观看图像:
130.
131.     #第二层输出 256 张 feature,这里显示 36 张。输入:
132.     feat = net.blobs['conv2'].data[4, :36]
133.     vis_square(feat, padval=1)
134.     #使用 ipt.show()观看图像
135.
136.     feat = net.blobs['conv2'].data[0, :36]
137.     vis_square(feat, padval=1)
138.
```

```
139. #第三个卷积层:全部 384 个 feature map,输入:
140. feat = net.blobs['conv3'].data[4]
141. vis_square(feat, padval=0.5)
142. #使用 ipt.show()查看图像:
143.
144. #第四个卷积层:全部 384 个 feature map,输入:
145. feat = net.blobs['conv4'].data[4]
146. vis_square(feat, padval=0.5)
147. #使用 ipt.show()查看图像:
148.
149. #第五个卷积层:全部 256 个 feature map,输入:
150. feat = net.blobs['conv5'].data[4]
151. vis_square(feat, padval=0.5)
152. #使用 ipt.show()查看图像:
153.
154. #第五个 pooling 层:我们也可以观察 pooling 层,输入:
155. feat = net.blobs['pool5'].data[4]
156. vis_square(feat, padval=1)
157. #使用 ipt.show()查看图像:
158.
159. #用caffe 的python接口提取和保存特征比较方便。
160. features = net.blobs['conv5'].data # 提取卷积层 5 的特征
161. np.savetxt('conv5_feature.txt', features) # 将特征存储到本文文件中
162.
163.
164. #然后我们看看第六层(第一个全连接层)输出后的直方分布:
165. feat = net.blobs['fc6'].data[4]
166. plt.subplot(2, 1, 1)
167. plt.plot(feat.flat)
168. plt.subplot(2, 1, 2)
169. _ = plt.hist(feat.flat[feat.flat > 0], bins=100)
170. #使用 ipt.show()查看图像:
171.
172. #第七层(第二个全连接层)输出后的直方分布:可以看出值的分布没有这么平均了。
173. feat = net.blobs['fc7'].data[4]
174. plt.subplot(2, 1, 1)
175. plt.plot(feat.flat)
176. plt.subplot(2, 1, 2)
```



```
177. _ = plt.hist(feats.flat[feats.flat > 0], bins=100)
178. #使用 plt.show() 查看图像:
179.
180. #The final probability output, prob
181. feat = net.blobs['prob'].data[0]
182. plt.plot(feats.flat)
183.
184.
185. #最后看看标签:Let's see the top 5 predicted labels.
186. # load labels
187. imagenet_labels_filename = caffe_root + 'data/ilsrvrc12/synset_words.txt'
188. try:
189.     labels = np.loadtxt(imagenet_labels_filename, str, delimiter='\t')
190. except:
191.     !../data/ilsrvrc12/get_ilsrvrc_aux.sh
192.     labels = np.loadtxt(imagenet_labels_filename, str, delimiter='\t')
193.
194. # sort top k predictions from softmax output
195. top_k = net.blobs['prob'].data[0].flatten().argsort()[-1:-6:-1]
196. print labels[top_k]
```

复制代码

备注：用 caffe 的 python 接口提取和保存特征到text文本下

```
01. features = net.blobs['conv5'].data # 提取卷积层 5 的特征
02. np.savetxt('conv5_feature.txt', features) # 将特征存储到本文文件中
```

复制代码

现在Caffe的Matlab接口 (matcaffe3) 和python接口都非常强大, 可以直接提取任意层的feature map以及parameters, 所以本文仅作为参考, 更多最新的信息请参考:

<http://caffe.berkeleyvision.org/tutorial/interfaces.html>

提取特征并储存

CAFFE提供了一个提取特征的tool, 见 <http://caffe.berkeleyvision.org/tutorial/extraction.html>。

数据模型与准备

安装好Caffe后, 在examples/images文件夹下有两张示例图像, 本文即在这两张图像上, 用Caffe提供的预训练模型, 进行特征提取, 并进行可视化。

1. 选择需要特征提取的图像

./examples/_temp

(1) 进入caffe根目录(本文中caffe的根目录都为caffe-root)，创建临时文件夹，用于存放所需要的临时文件

```
01.  mkdir examples/_temp
```

复制代码

(2) 根据examples/images文件夹中的图片，创建包含图像列表的txt文件，并添加标签（0）

```
01.  find `pwd`/examples/images -type f -exec echo {} \; > examples/_temp/temp.txt
```

```
02.  sed "s/$/ 0/" examples/_temp/temp.txt > examples/_temp/file_list.txt
```

复制代码

(3) 执行下列脚本，下载imagenet12图像均值文件，在后面的网络结构定义prototxt文件中，需要用到该文件（data/ilsrvr212/imagenet_mean.binaryproto）。下载模型以及定义prototxt。

```
01.  sh ./data/ilsrvr12/get_ilsrvr_aux.sh
```

复制代码

(4) 将网络定义prototxt文件复制到_temp文件夹下

```
01.  cp examples/feature_extraction/imagenet_val.prototxt examples/_temp
```

复制代码

使用特征文件进行可视化

参考 <http://www.cnblogs.com/platero/p/3967208.html> 和 [lmdb](https://lmdb.readthedocs.org/en/release) 的文档 <https://lmdb.readthedocs.org/en/release>，读取lmdb文件，然后转换成mat文件，再用matlab调用mat进行可视化。

使用caffe的 extract_features.bin 工具提取出的图像特征存为lmdb格式，为了方便观察特征，我们将利用下列两个python脚本将图像转化为matlab的.mat格式（请先安装caffe的python依赖库）。extract_features.bin的运行参数为

```
01.  extract_features.bin $MODEL $PROTOTXT $LAYER $LMDB_OUTPUT_PATH $BATCHSIZE
```

复制代码

上面不是执行代码，只是运行参数，不需要执行上式。

下面给出第一个例子是提取特征并储存。

（1）安装CAFFE的python依赖库，并使用以下两个辅助文件把lmdb转换为mat。在caffe 根目录下创建feat_helper_pb2.py和lmdb2mat.py，直接copy 下面的python程序即可。

```
01. cd caffe-root
02. sudo gedit feat_helper_pb2.py
03. sudo gedit lmdb2mat.py
```

复制代码

需要添加的内容如下

feat_helper_pb2.py：

```
01. # Generated by the protocol buffer compiler.  DO NOT EDIT!
02.
03. from google.protobuf import descriptor
04. from google.protobuf import message
05. from google.protobuf import reflection
06. from google.protobuf import descriptor_pb2
07. # @@protoc_insertion_point(imports)
08.
09.
10. DESCRIPTOR = descriptor.FileDescriptor(
11.     name='datum.proto',
12.     package='feat_extract',
13.     serialized_pb='\n\x0b\x64\x61tum.proto\x12\x0c\x66\x65\x61t_extract\"i\n\x05\x44\x61tum\x12\x10\n\x08\x63annels\x18\x01 \x01(\x05\x12\x0e\n\x06height\x18\x02 \x01(\x05\x12\r\n\x05width\x18\x03 \x01(\x05\x12\x0c\n\x04\x64\x61ta\x18\x04 \x01(\x0c\x12\r\n\x05label\x18\x05 \x01(\x05\x12\x12\n\nfloat_data\x18\x06 \x03(\x02')
14.
15.
16. _DATUM = descriptor.Descriptor(
17.     name='Datum',
18.     full_name='feat_extract.Datum',
19.     filename=None,
20.     file=DESCRIPTOR,
21.     containing_type=None,
22.     fields=[
23.         descriptor.FieldDescriptor(
```

```
24.     name='channels', full_name='feat_extract.Datum.channels', index=0,
25.     number=1, type=5, cpp_type=1, label=1,
26.     has_default_value=False, default_value=0,
27.     message_type=None, enum_type=None, containing_type=None,
28.     is_extension=False, extension_scope=None,
29.     options=None),
30. descriptor.FieldDescriptor(
31.     name='height', full_name='feat_extract.Datum.height', index=1,
32.     number=2, type=5, cpp_type=1, label=1,
33.     has_default_value=False, default_value=0,
34.     message_type=None, enum_type=None, containing_type=None,
35.     is_extension=False, extension_scope=None,
36.     options=None),
37. descriptor.FieldDescriptor(
38.     name='width', full_name='feat_extract.Datum.width', index=2,
39.     number=3, type=5, cpp_type=1, label=1,
40.     has_default_value=False, default_value=0,
41.     message_type=None, enum_type=None, containing_type=None,
42.     is_extension=False, extension_scope=None,
43.     options=None),
44. descriptor.FieldDescriptor(
45.     name='data', full_name='feat_extract.Datum.data', index=3,
46.     number=4, type=12, cpp_type=9, label=1,
47.     has_default_value=False, default_value="",
48.     message_type=None, enum_type=None, containing_type=None,
49.     is_extension=False, extension_scope=None,
50.     options=None),
51. descriptor.FieldDescriptor(
52.     name='label', full_name='feat_extract.Datum.label', index=4,
53.     number=5, type=5, cpp_type=1, label=1,
54.     has_default_value=False, default_value=0,
55.     message_type=None, enum_type=None, containing_type=None,
56.     is_extension=False, extension_scope=None,
57.     options=None),
58. descriptor.FieldDescriptor(
59.     name='float_data', full_name='feat_extract.Datum.float_data', index=5,
60.     number=6, type=2, cpp_type=6, label=3,
61.     has_default_value=False, default_value=[],
```

```
62.         message_type=None, enum_type=None, containing_type=None,
63.         is_extension=False, extension_scope=None,
64.         options=None),
65.     ],
66.     extensions=[
67.     ],
68.     nested_types=[],
69.     enum_types=[
70.     ],
71.     options=None,
72.     is_extendable=False,
73.     extension_ranges=[],
74.     serialized_start=29,
75.     serialized_end=134,
76. )
77.
78. DESCRIPTOR.message_types_by_name['Datum'] = _DATUM
79.
80. class Datum(message.Message):
81.     __metaclass__ = reflection.GeneratedProtocolMessageType
82.     DESCRIPTOR = _DATUM
83.
84.     # @@protoc_insertion_point(class_scope:feat_extract.Datum)
85.
86.     # @@protoc_insertion_point(module_scope)
```

复制代码

/lmbd2mat.py

```
01. import lmbd
02. import feat_helper_pb2
03. import numpy as np
04. import scipy.io as sio
05. import time
06.
07. def main(argv):
08.     lmbd_name = sys.argv[1]
09.     print "%s" % sys.argv[1]
```

```
10.     batch_num = int(sys.argv[2]);
11.     batch_size = int(sys.argv[3]);
12.     window_num = batch_num*batch_size;
13.
14.     start = time.time()
15.     if 'db' not in locals().keys():
16.         db = lmdb.open(lmdb_name)
17.         txn= db.begin()
18.         cursor = txn.cursor()
19.         cursor.iternext()
20.         datum = feat_helper_pb2.Datum()
21.
22.         keys = []
23.         values = []
24.         for key, value in enumerate( cursor.iternext_nodup()):
25.             keys.append(key)
26.             values.append(cursor.value())
27.
28.         ft = np.zeros((window_num, int(sys.argv[4])))
29.         for im_idx in range(window_num):
30.             datum.ParseFromString(values[im_idx])
31.             ft[im_idx, :] = datum.float_data
32.
33.         print 'time 1: %f' %(time.time() - start)
34.         sio.savemat(sys.argv[5], {'feats':ft})
35.         print 'time 2: %f' %(time.time() - start)
36.         print 'done!'
37.
38. if __name__ == '__main__':
39.     import sys
40.     main(sys.argv)
```

复制代码

备注：用 caffe 的 python 接口提取和保存特征到text文本下

```
01. features = net.blobs['conv5'].data # 提取卷积层 5 的特征
02. np.savetxt('conv5_feature.txt', features) # 将特征存储到本文文件中
```

[复制代码](#)

（2）在caffe 根目录下创建脚本文件extract_feature_example1.sh，并执行，将在examples/_temp文件夹下得到lmdb文件（features_conv1）和.mat文件（features_conv1.mat）

下载已经生成的模型

```
01. sudo gedit ./examples/imagenet/get_caffe_reference_imagenet_model.sh
```

[复制代码](#)

添加编辑内容如下：

```
01.  #!/usr/bin/env sh
02.  # This scripts downloads the caffe reference imagenet model
03.  # for ilsvrc image classification and deep feature extraction
04.
05.  MODEL=caffe_reference_imagenet_model
06.  CHECKSUM=bf44bac4a59aa7792b296962fe483f2b
07.
08.  if [ -f $MODEL ]; then
09.      echo "Model already exists. Checking md5..."
10.      os=`uname -s`
11.      if [ "$os" = "Linux" ]; then
12.          checksum=`md5sum $MODEL | awk '{ print $1 }'`
13.      elif [ "$os" = "Darwin" ]; then
14.          checksum=`cat $MODEL | md5`
15.      fi
16.      if [ "$checksum" = "$CHECKSUM" ]; then
17.          echo "Model checksum is correct. No need to download."
18.          exit 0
19.      else
20.          echo "Model checksum is incorrect. Need to download again."
21.      fi
22.  fi
23.
24.  echo "Downloading..."
25.
26.  wget --no-check-certificate https://www.dropbox.com/s/n3jups0gr7uj0dv/$MODEL
```

```
27.  
28.  echo "Done. Please run this command again to verify that checksum = $CHECKSUM."
```

复制代码

```
01.  cd caffe-root  
02.  sudo gedit extract_feature_example1.sh
```

复制代码

```
01.  #!/usr/bin/env sh  
02.  # args for EXTRACT_FEATURE  
03.  TOOL=./build/tools  
04.  MODEL=./examples/imagenet/caffe_reference_imagenet_model #下载得到的caffe model  
05.  PROTOTXT=./examples/_temp/imagenet_val.prototxt # 网络定义  
06.  LAYER=conv1 # 提取层的名字，如提取fc7等  
07.  LEVELDB=./examples/_temp/features_conv1 # 保存的leveldb路径  
08.  BATCHSIZE=10  
09.  
10.  # args for LEVELDB to MAT  
11.  DIM=290400 # 需要手工计算feature长度  
12.  OUT=./examples/_temp/features_conv1.mat #.mat文件保存路径  
13.  BATCHNUM=1 # 有多少个batch， 本例只有两张图， 所以只有一个batch  
14.  
15.  $TOOL/extract_features.bin $MODEL $PROTOTXT $LAYER $LEVELDB $BATCHSIZE lmbd  
16.  python lmbd2mat.py $LEVELDB $BATCHNUM $BATCHSIZE $DIM $OUT
```

复制代码

执行之后，

```
01.  cd caffe-root  
02.  sh extract_feature_example1.sh
```

复制代码

你会在/examples/_temp/ 下发现多了两个文件：文件夹 features_conv1，文件features_conv1.mat



如果执行出现lmbd2mat.py的相关问题，有可能是没有安装lmbd，可在caffe 根目录下执行下面的程式安装。具体问题具体分析。

Lmdb的安装

```
01. pip install lmdb
```

复制代码

转自<http://blog.csdn.net/jiandanjinxin/article/details/50410290>

分享到: QQ好友和群 QQ空间 腾讯微博 腾讯朋友

收藏 转播 支持 反对

举报

回复

返回列表

高级模式


您需要登录后才可以回帖 登录 | 注册nanjixiong2017册  用QQ帐号登录

发表回复

☐ 回帖并转播

☐ 回帖后跳转到最后一页

本版积分规则



关于我们

联系我们

招聘信息

其他平台

3D打印微信

3D打印微博

3D打印培训

导航

熊玩意3D打印模型

3D打印导航

3D打印入门