[学习笔记：Caffe上LeNet模型理解](http://blog.csdn.net/lynnandwei/article/details/44082859)

分类： [Machine Learning](http://blog.csdn.net/lynnandwei/article/category/2864799)2015-03-05 17:23 377人阅读 [评论](http://blog.csdn.net/lynnandwei/article/details/44082859#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/lynnandwei/article/details/44082859#report)

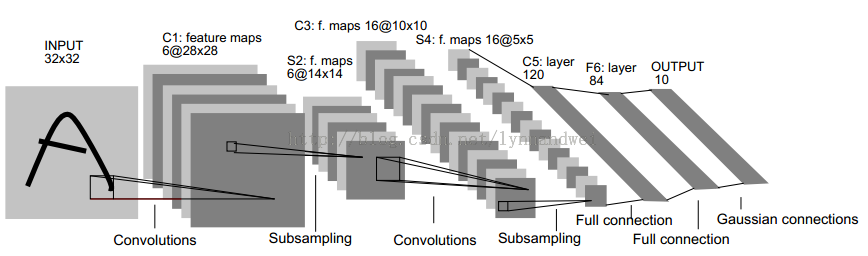
Caffe中用的模型结构是著名的手写体识别模型LeNet-5（http://yann.lecun.com/exdb/lenet/a35.html）。当年美国大多数银行就是用它来识别支票上面的手写数字的。能够达到这种商用的地步，它的准确性可想而知，唯一的区别是把其中的sigmoid激活函数换成了ReLU。

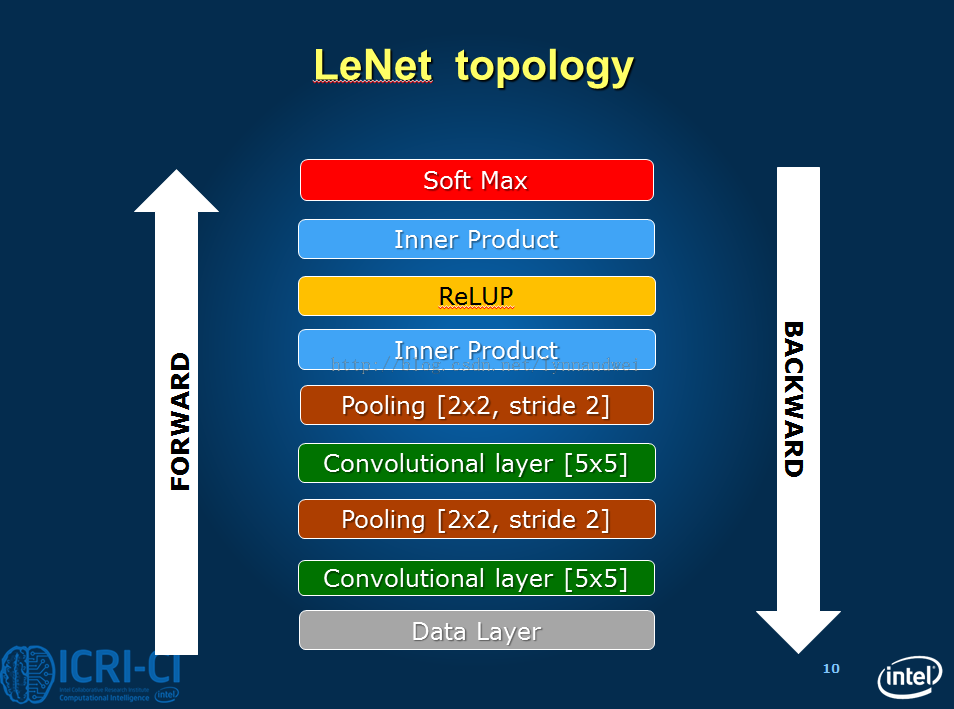
为什么换成ReLU，上一篇blog中找到了一些相关讨论，可以参考。

CNN的发展，关键就在于，通过卷积（convolution http://deeplearning.stanford.edu/wiki/index.php/Feature\_extraction\_using\_convolution）和降采样（pooling http://deeplearning.stanford.edu/wiki/index.php/Pooling ）能够成功的减少需要训练的参数值，回头去看SparseAutoEncoder 更会有明显的感觉。

具体需要训练多少个参数，http://blog.csdn.net/zouxy09/article/details/8781543 有做一个对应的推算，可以参考。

这是一个原始的LeNet模型图





在Caffe中，这个结构进行了一些修改。结构定义在$caffe-master/examples/mnist/lenet\_train\_test.prototxt中。

需要对google protobuf有一定了解并且看过Caffe中protobuf的定义，其定义在$caffe-master/src/caffe/proto/caffe.proto。  
protobuf是google公司的一个开源项目，主要功能是把某种数据结构的信息以某种格式保存及传递，类似微软的XML，但是效率较高。目前提供C++、java和python的API。  
protobuf简介：http://blog.163.com/jiang\_tao\_2010/blog/static/12112689020114305013458/  
使用实例       ：http://www.ibm.com/developerworks/cn/linux/l-cn-gpb/

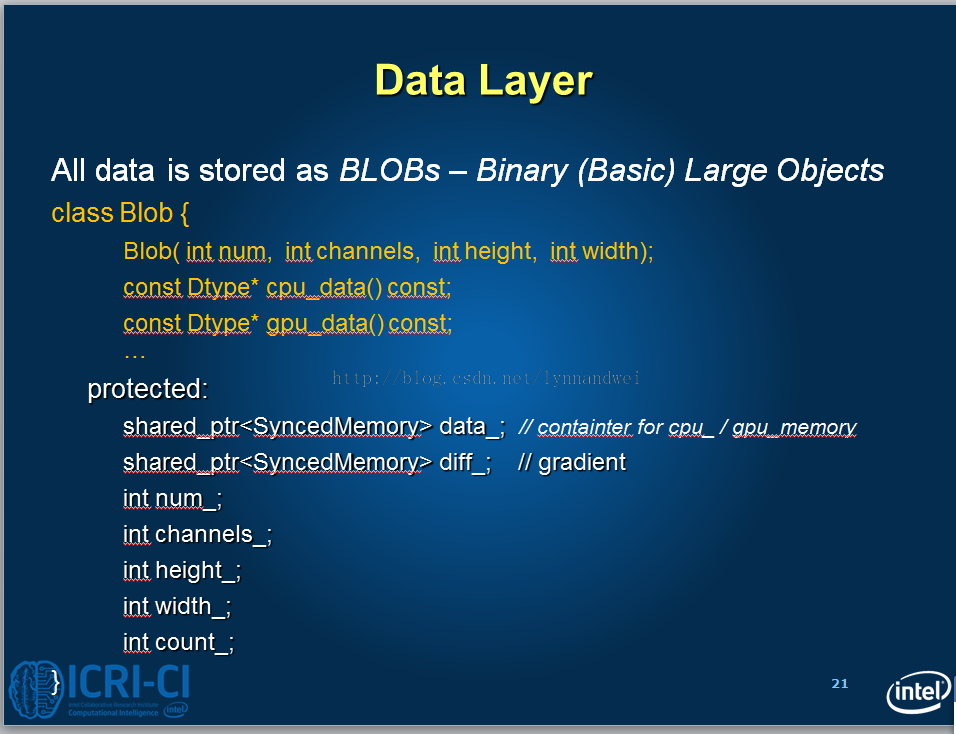
Blob

Blob是用以存储数据的4维数组，例如

对于数据：Number\*Channel\*Height\*Width

对于卷积权重：Output\*Input\*Height\*Width

对于卷积偏置：Output\*1\*1\*1



整个结构中包含两个convolution layer、两个pooling layer和两个fully connected layer。

每个层有多个Feature Map，每个Feature Map通过一种卷积滤波器提取输入的一种特征，然后每个Feature Map有多个神经元。  
  
首先是数据层，测试数据100张为一批（batch\_size），后面括号内是数据总大小。如100\*28\*28= 78400

 Top shape: 100 1 28 28 (78400)

 Top shape: 100 1 1 1 (100)

conv1(即产生图上 C1数据）层是一个卷积层，由20个特征图Feature Map构成。卷积核的大小是5\*5。 通过卷积之后，数据变成（28-5+1）\*（28-5+1），20个特征

我们是可以随机的初始化权重和偏差，使用xavier算法根据输入和输出的神经元数目来决定初始化的范围。

Top shape: 100 20 24 24 (1152000)

pool1（即产生S2数据）是一个降采样层，有20个12\*12的特征图。降采样的核是2\*2的，所以数据变成12\*12.

Top shape: 100 20 12 12 (288000)

conv2（即产生C3数据）是卷积层，核还是5\*5，数据变成（12-5+1）\*（12-5+1）。 50个特征

Top shape: 100 50 8 8 (320000)    
  
pool2（即产生S3数据）是降采样层，降采样核为2\*2，则数据变成4\*4  
  
 Top shape: 100 50 4 4 (80000)

ip1 是全连接层（产生C5的数据）。某个程度上可以认为是卷积层。输出为500. 原始模型中，从5\*5的数据通过5\*5的卷积得到1\*1的数据。 现在的模型数据为4\*4，得到的数据也是1\*1，构成了数据中的全连接。

Top shape: 100 500 1 1 (50000)

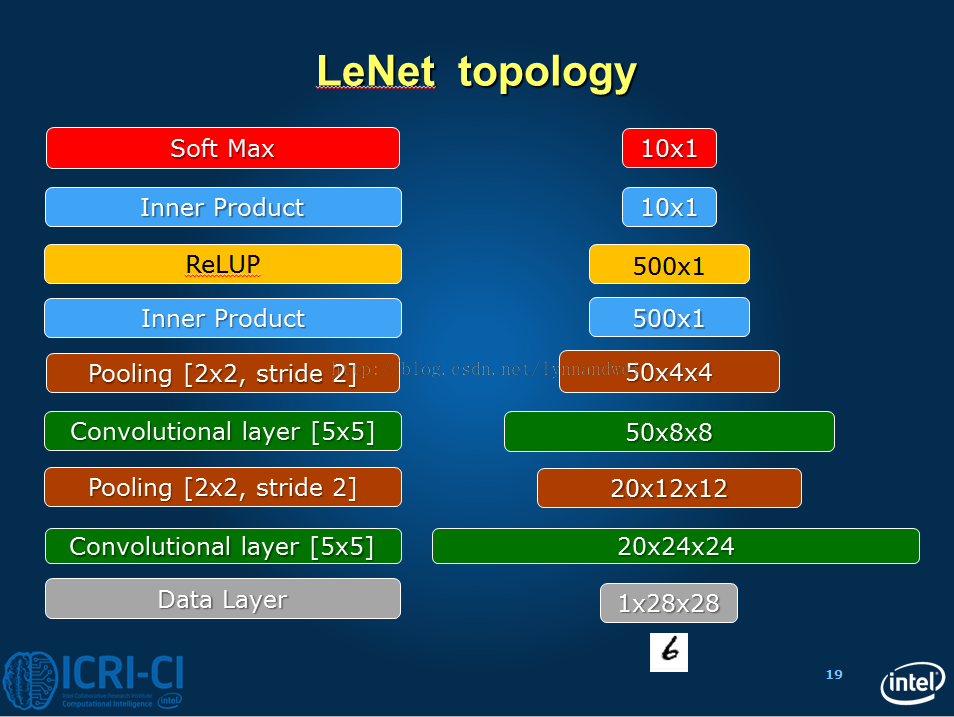
通过RELU 计算

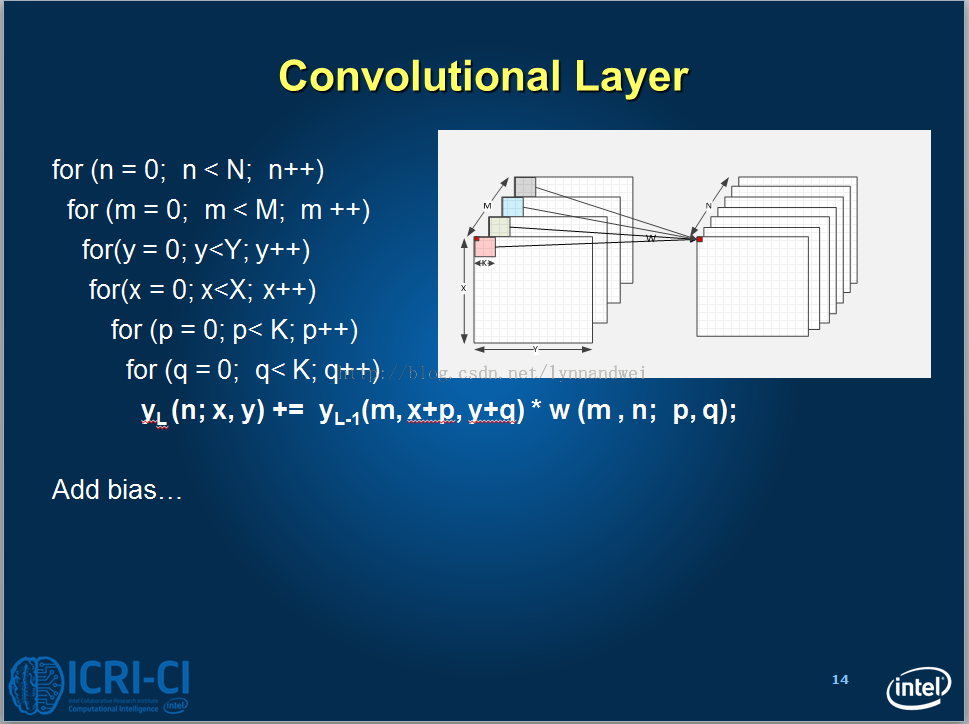
Top shape: 100 500 1 1 (50000)

ip2是第二个全连接层，输出为10，直接输出结果，数据的分类判断在这一层中完成。

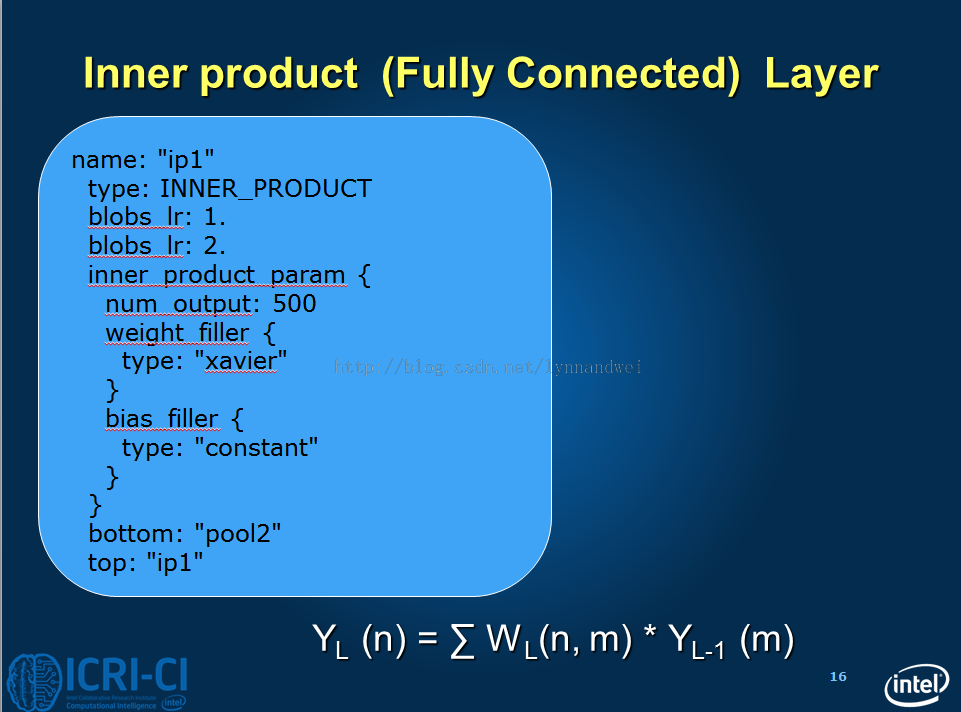
1. I0303 18:26:32.104604 27313 net.cpp:96] Setting up ip2
2. I0303 18:26:32.104676 27313 net.cpp:103] Top shape: 100 10 1 1 (1000)
3. I0303 18:26:32.104691 27313 net.cpp:67] Creating Layer ip2\_ip2\_0\_split
4. I0303 18:26:32.104701 27313 net.cpp:394] ip2\_ip2\_0\_split **<-** ip2
5. I0303 18:26:32.104710 27313 net.cpp:356] ip2\_ip2\_0\_split -**>** ip2\_ip2\_0\_split\_0
6. I0303 18:26:32.104722 27313 net.cpp:356] ip2\_ip2\_0\_split -**>** ip2\_ip2\_0\_split\_1
7. I0303 18:26:32.104733 27313 net.cpp:96] Setting up ip2\_ip2\_0\_split
8. I0303 18:26:32.104743 27313 net.cpp:103] Top shape: 100 10 1 1 (1000)

 Top shape: 100 10 1 1 (1000)    
  
数据变化对比如图

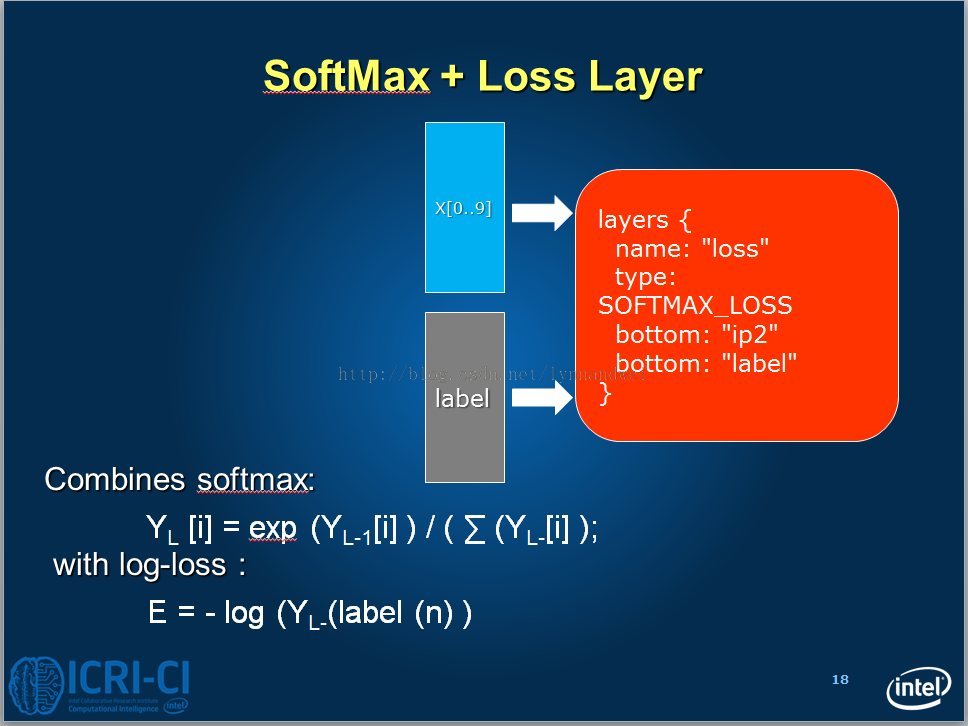
  
  
  
  
此外，从pool1到conv2， 整个过程应该是怎样的，也可以用图来表示，其中m=20, n = 50 x=y=12, k=5



ip1 虽然这一层有其他的数据操作，但是最终可以用如下的公式来进行计算。所以它也是全连接层



loss的公式



整个网络的反向求导具体如下：

资料可参照 http://blog.csdn.net/zouxy09/article/details/9993371  http://www.cnblogs.com/tornadomeet/p/3468450.html

1. I0303 18:26:32.104909 27313 net.cpp:170] loss needs backward computation.
2. I0303 18:26:32.104918 27313 net.cpp:172] accuracy does not need backward computation.
3. I0303 18:26:32.104925 27313 net.cpp:170] ip2\_ip2\_0\_split needs backward computation.
4. I0303 18:26:32.104933 27313 net.cpp:170] ip2 needs backward computation.
5. I0303 18:26:32.104941 27313 net.cpp:170] relu1 needs backward computation.
6. I0303 18:26:32.104948 27313 net.cpp:170] ip1 needs backward computation.
7. I0303 18:26:32.104956 27313 net.cpp:170] pool2 needs backward computation.
8. I0303 18:26:32.104964 27313 net.cpp:170] conv2 needs backward computation.
9. I0303 18:26:32.104975 27313 net.cpp:170] pool1 needs backward computation.
10. I0303 18:26:32.104984 27313 net.cpp:170] conv1 needs backward computation.

参考文献

机器学习(Machine Learning)&深度学习(Deep Learning)资料  http://blog.csdn.net/zhoubl668/article/details/42921187

http://ml.memect.com/article/machine-learning-guide.html

http://www.cnblogs.com/tornadomeet/p/3468450.html

http://www.360doc.com/content/13/0729/19/13256259\_303401668.shtml   
http://blog.sciencenet.cn/blog-1583812-843207.html  
http://blog.csdn.net/qiaofangjie/article/details/16826849

http://blog.csdn.net/zouxy09/article/details/9993371

http://www.cnblogs.com/tornadomeet/archive/2013/05/05/3061457.html

http://blog.csdn.net/kkk584520/article/details/41694301

http://blog.csdn.net/ycheng\_sjtu/article/details/39693655

[学习笔记-CIFAR10模型理解简述](http://blog.csdn.net/lynnandwei/article/details/44302175)

分类： [Machine Learning](http://blog.csdn.net/lynnandwei/article/category/2864799)2015-03-16 10:53 297人阅读 [评论](http://blog.csdn.net/lynnandwei/article/details/44302175#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/lynnandwei/article/details/44302175#report)

整个结构中包含三个convolution layer、三个pooling layer和两个fully connected layer。

每个层有多个Feature Map，每个Feature Map通过一种卷积滤波器提取输入的一种特征，然后每个Feature Map有多个神经元。  
  
首先是数据层，测试数据100张为一批（batch\_size），后面括号内是数据总大小。如100\*32\*32\*3= 307200

 Top shape: 100 3 32 32 (307200)

 Top shape: 100 1 1 1 (100)

conv1(即产生图上 C1数据）层是一个卷积层，由32个特征图Feature Map构成。卷积核的大小是5\*5,因为有pad为2，也就是每边增加两个单位的边界。 通过卷积之后，数据变成（32+2\*2-5+1）\*（32+2\*2-5+1）

1. layers {
2. name: "conv1"
3. type: CONVOLUTION
4. bottom: "data"
5. top: "conv1"
6. blobs\_lr: 1
7. blobs\_lr: 2
8. convolution\_param {
9. num\_output: 32
10. pad: 2
11. kernel\_size: 5
12. stride: 1
13. weight\_filler {
14. type: "gaussian"
15. std: 0.0001
16. }
17. bias\_filler {
18. type: "constant"
19. }
20. }
21. }

Top shape: 100 32 32 32 (3276800)

pool1 是一个降采样层，有32个16\*16的特征图。降采样的核是2\*2的，所以数据变成16\*16.

Top shape: 100 32 16 16 (819200)

然后接入RELU1

Top shape: 100 32 16 16 (819200)

conv2 是卷积层，核还是5\*5，pad还是2。

Top shape: 100 32 16 16 (819200)     
  
然后接入RELU2

Top shape: 100 32 16 16 (819200)

pool2是降采样层，降采样核为2\*2，则数据变成8\*8

Top shape: 100 32 8 8 (204800)

conv3 是卷积层，核还是5\*5，pad还是2，特征是64个。

**[html]** [view plaincopy](http://blog.csdn.net/lynnandwei/article/details/44302175)[在CODE上查看代码片](https://code.csdn.net/snippets/620442)

1. Top shape: 100 64 8 8 (409600)

然后接入RELU2

Top shape: 100 64 8 8 (409600)

pool3是降采样层，降采样核为2\*2，则数据变成4\*4

**[html]** [view plaincopy](http://blog.csdn.net/lynnandwei/article/details/44302175)[在CODE上查看代码片](https://code.csdn.net/snippets/620442)

1. Top shape: 100 64 4 4 (102400)

ip1 是全连接层。某个程度上可以认为是卷积层。输出为64. 原始模型中，从5\*5的数据通过5\*5的卷积得到1\*1的数据。 现在的模型数据为4\*4，得到的数据也是1\*1，构成了数据中的全连接。

Top shape: 100 64 1 1 (6400)

ip2是第二个全连接层，输出为10，直接输出结果，数据的分类判断在这一层中完成。

Top shape: 100 64 8 8 (409600)

**[html]** [view plaincopy](http://blog.csdn.net/lynnandwei/article/details/44302175)[在CODE上查看代码片](https://code.csdn.net/snippets/620442)

1. I0313 00:40:24.570014 13825 net.cpp:96] Setting up ip2
2. I0313 00:40:24.570108 13825 net.cpp:103] Top shape: 100 10 1 1 (1000)
3. I0313 00:40:24.570114 13825 net.cpp:113] Memory required for data: 31979600
4. I0313 00:40:24.570127 13825 net.cpp:67] Creating Layer ip2\_ip2\_0\_split
5. I0313 00:40:24.570134 13825 net.cpp:394] ip2\_ip2\_0\_split **<-** ip2
6. I0313 00:40:24.570143 13825 net.cpp:356] ip2\_ip2\_0\_split -**>** ip2\_ip2\_0\_split\_0
7. I0313 00:40:24.570154 13825 net.cpp:356] ip2\_ip2\_0\_split -**>** ip2\_ip2\_0\_split\_1
8. I0313 00:40:24.570163 13825 net.cpp:96] Setting up ip2\_ip2\_0\_split
9. I0313 00:40:24.570171 13825 net.cpp:103] Top shape: 100 10 1 1 (1000)
10. I0313 00:40:24.570176 13825 net.cpp:103] Top shape: 100 10 1 1 (1000)
11. I0313 00:40:24.570181 13825 net.cpp:113] Memory required for data: 31987600

输入猫的图片



输出结果为：

['deer' 'airplane' 'cat' 'frog' 'bird']

**[html]** [view plaincopy](http://blog.csdn.net/lynnandwei/article/details/44302175)[在CODE上查看代码片](https://code.csdn.net/snippets/620442)

1. I0313 00:40:24.471560 13825 net.cpp:67] Creating Layer cifar
2. I0313 00:40:24.471570 13825 net.cpp:356] cifar -**>** data
3. I0313 00:40:24.471585 13825 net.cpp:356] cifar -**>** label
4. I0313 00:40:24.471596 13825 net.cpp:96] Setting up cifar
5. I0313 00:40:24.471602 13825 data\_layer.cpp:45] Opening leveldb examples/cifar10/cifar10\_test\_leveldb
6. I0313 00:40:24.549324 13825 data\_layer.cpp:128] output data size: 100,3,32,32
7. I0313 00:40:24.549372 13825 base\_data\_layer.cpp:36] Loading mean file fromexamples/cifar10/mean.binaryproto
8. I0313 00:40:24.550582 13825 base\_data\_layer.cpp:64] Initializing prefetch
9. I0313 00:40:24.550639 13825 base\_data\_layer.cpp:66] Prefetch initialized.
10. I0313 00:40:24.550683 13825 net.cpp:103] Top shape: 100 3 32 32 (307200)
11. I0313 00:40:24.550698 13825 net.cpp:103] Top shape: 100 1 1 1 (100)
12. I0313 00:40:24.550709 13825 net.cpp:113] Memory required for data: 1229200
13. I0313 00:40:24.550734 13825 net.cpp:67] Creating Layer label\_cifar\_1\_split
14. I0313 00:40:24.550750 13825 net.cpp:394] label\_cifar\_1\_split **<-** label
15. I0313 00:40:24.550775 13825 net.cpp:356] label\_cifar\_1\_split -**>** label\_cifar\_1\_split\_0
16. I0313 00:40:24.550802 13825 net.cpp:356] label\_cifar\_1\_split -**>** label\_cifar\_1\_split\_1
17. I0313 00:40:24.550824 13825 net.cpp:96] Setting up label\_cifar\_1\_split
18. I0313 00:40:24.550843 13825 net.cpp:103] Top shape: 100 1 1 1 (100)
19. I0313 00:40:24.550855 13825 net.cpp:103] Top shape: 100 1 1 1 (100)
20. I0313 00:40:24.550866 13825 net.cpp:113] Memory required for data: 1230000
21. I0313 00:40:24.550889 13825 net.cpp:67] Creating Layer conv1
22. I0313 00:40:24.550902 13825 net.cpp:394] conv1 **<-** data
23. I0313 00:40:24.550926 13825 net.cpp:356] conv1 -**>** conv1
24. I0313 00:40:24.550951 13825 net.cpp:96] Setting up conv1
25. I0313 00:40:24.551573 13825 net.cpp:103] Top shape: 100 32 32 32 (3276800)
26. I0313 00:40:24.551583 13825 net.cpp:113] Memory required for data: 14337200
27. I0313 00:40:24.551599 13825 net.cpp:67] Creating Layer pool1
28. I0313 00:40:24.551605 13825 net.cpp:394] pool1 **<-** conv1
29. I0313 00:40:24.551615 13825 net.cpp:356] pool1 -**>** pool1
30. I0313 00:40:24.551625 13825 net.cpp:96] Setting up pool1
31. I0313 00:40:24.551633 13825 net.cpp:103] Top shape: 100 32 16 16 (819200)
32. I0313 00:40:24.551638 13825 net.cpp:113] Memory required for data: 17614000
33. I0313 00:40:24.551652 13825 net.cpp:67] Creating Layer relu1
34. I0313 00:40:24.551658 13825 net.cpp:394] relu1 **<-** pool1
35. I0313 00:40:24.551667 13825 net.cpp:345] relu1 -**>** pool1 (in-place)
36. I0313 00:40:24.551676 13825 net.cpp:96] Setting up relu1
37. I0313 00:40:24.551682 13825 net.cpp:103] Top shape: 100 32 16 16 (819200)
38. I0313 00:40:24.551687 13825 net.cpp:113] Memory required for data: 20890800
39. I0313 00:40:24.551695 13825 net.cpp:67] Creating Layer conv2
40. I0313 00:40:24.551700 13825 net.cpp:394] conv2 **<-** pool1
41. I0313 00:40:24.551710 13825 net.cpp:356] conv2 -**>** conv2
42. I0313 00:40:24.551720 13825 net.cpp:96] Setting up conv2
43. I0313 00:40:24.554986 13825 net.cpp:103] Top shape: 100 32 16 16 (819200)
44. I0313 00:40:24.554996 13825 net.cpp:113] Memory required for data: 24167600
45. I0313 00:40:24.555009 13825 net.cpp:67] Creating Layer relu2
46. I0313 00:40:24.555024 13825 net.cpp:394] relu2 **<-** conv2
47. I0313 00:40:24.555034 13825 net.cpp:345] relu2 -**>** conv2 (in-place)
48. I0313 00:40:24.555043 13825 net.cpp:96] Setting up relu2
49. I0313 00:40:24.555049 13825 net.cpp:103] Top shape: 100 32 16 16 (819200)
50. I0313 00:40:24.555054 13825 net.cpp:113] Memory required for data: 27444400
51. I0313 00:40:24.555061 13825 net.cpp:67] Creating Layer pool2
52. I0313 00:40:24.555068 13825 net.cpp:394] pool2 **<-** conv2
53. I0313 00:40:24.555076 13825 net.cpp:356] pool2 -**>** pool2
54. I0313 00:40:24.555085 13825 net.cpp:96] Setting up pool2
55. I0313 00:40:24.555094 13825 net.cpp:103] Top shape: 100 32 8 8 (204800)
56. I0313 00:40:24.555099 13825 net.cpp:113] Memory required for data: 28263600
57. I0313 00:40:24.555109 13825 net.cpp:67] Creating Layer conv3
58. I0313 00:40:24.555114 13825 net.cpp:394] conv3 **<-** pool2
59. I0313 00:40:24.555124 13825 net.cpp:356] conv3 -**>** conv3
60. I0313 00:40:24.555135 13825 net.cpp:96] Setting up conv3
61. I0313 00:40:24.561589 13825 net.cpp:103] Top shape: 100 64 8 8 (409600)
62. I0313 00:40:24.561599 13825 net.cpp:113] Memory required for data: 29902000
63. I0313 00:40:24.561611 13825 net.cpp:67] Creating Layer relu3
64. I0313 00:40:24.561619 13825 net.cpp:394] relu3 **<-** conv3
65. I0313 00:40:24.561627 13825 net.cpp:345] relu3 -**>** conv3 (in-place)
66. I0313 00:40:24.561636 13825 net.cpp:96] Setting up relu3
67. I0313 00:40:24.561642 13825 net.cpp:103] Top shape: 100 64 8 8 (409600)
68. I0313 00:40:24.561646 13825 net.cpp:113] Memory required for data: 31540400
69. I0313 00:40:24.561655 13825 net.cpp:67] Creating Layer pool3
70. I0313 00:40:24.561661 13825 net.cpp:394] pool3 **<-** conv3
71. I0313 00:40:24.561669 13825 net.cpp:356] pool3 -**>** pool3
72. I0313 00:40:24.561678 13825 net.cpp:96] Setting up pool3
73. I0313 00:40:24.561686 13825 net.cpp:103] Top shape: 100 64 4 4 (102400)
74. I0313 00:40:24.561691 13825 net.cpp:113] Memory required for data: 31950000
75. I0313 00:40:24.561699 13825 net.cpp:67] Creating Layer ip1
76. I0313 00:40:24.561704 13825 net.cpp:394] ip1 **<-** pool3
77. I0313 00:40:24.561714 13825 net.cpp:356] ip1 -**>** ip1
78. I0313 00:40:24.561724 13825 net.cpp:96] Setting up ip1
79. I0313 00:40:24.569967 13825 net.cpp:103] Top shape: 100 64 1 1 (6400)
80. I0313 00:40:24.569975 13825 net.cpp:113] Memory required for data: 31975600
81. I0313 00:40:24.569988 13825 net.cpp:67] Creating Layer ip2
82. I0313 00:40:24.569993 13825 net.cpp:394] ip2 **<-** ip1
83. I0313 00:40:24.570004 13825 net.cpp:356] ip2 -**>** ip2
84. I0313 00:40:24.570014 13825 net.cpp:96] Setting up ip2
85. I0313 00:40:24.570108 13825 net.cpp:103] Top shape: 100 10 1 1 (1000)
86. I0313 00:40:24.570114 13825 net.cpp:113] Memory required for data: 31979600
87. I0313 00:40:24.570127 13825 net.cpp:67] Creating Layer ip2\_ip2\_0\_split
88. I0313 00:40:24.570134 13825 net.cpp:394] ip2\_ip2\_0\_split **<-** ip2
89. I0313 00:40:24.570143 13825 net.cpp:356] ip2\_ip2\_0\_split -**>** ip2\_ip2\_0\_split\_0
90. I0313 00:40:24.570154 13825 net.cpp:356] ip2\_ip2\_0\_split -**>** ip2\_ip2\_0\_split\_1
91. I0313 00:40:24.570163 13825 net.cpp:96] Setting up ip2\_ip2\_0\_split
92. I0313 00:40:24.570171 13825 net.cpp:103] Top shape: 100 10 1 1 (1000)
93. I0313 00:40:24.570176 13825 net.cpp:103] Top shape: 100 10 1 1 (1000)
94. I0313 00:40:24.570181 13825 net.cpp:113] Memory required for data: 31987600
95. I0313 00:40:24.570189 13825 net.cpp:67] Creating Layer accuracy
96. I0313 00:40:24.570194 13825 net.cpp:394] accuracy **<-** ip2\_ip2\_0\_split\_0
97. I0313 00:40:24.570202 13825 net.cpp:394] accuracy **<-** label\_cifar\_1\_split\_0
98. I0313 00:40:24.570214 13825 net.cpp:356] accuracy -**>** accuracy
99. I0313 00:40:24.570222 13825 net.cpp:96] Setting up accuracy
100. I0313 00:40:24.570230 13825 net.cpp:103] Top shape: 1 1 1 1 (1)
101. I0313 00:40:24.570235 13825 net.cpp:113] Memory required for data: 31987604
102. I0313 00:40:24.570245 13825 net.cpp:67] Creating Layer loss
103. I0313 00:40:24.570250 13825 net.cpp:394] loss **<-** ip2\_ip2\_0\_split\_1
104. I0313 00:40:24.570257 13825 net.cpp:394] loss **<-** label\_cifar\_1\_split\_1
105. I0313 00:40:24.570266 13825 net.cpp:356] loss -**>** loss
106. I0313 00:40:24.570274 13825 net.cpp:96] Setting up loss
107. I0313 00:40:24.570286 13825 net.cpp:103] Top shape: 1 1 1 1 (1)
108. I0313 00:40:24.570291 13825 net.cpp:109]     with loss weight 1
109. I0313 00:40:24.570305 13825 net.cpp:113] Memory required for data: 31987608
110. I0313 00:40:24.570312 13825 net.cpp:170] loss needs backward computation.
111. I0313 00:40:24.570317 13825 net.cpp:172] accuracy does not need backward computation.
112. I0313 00:40:24.570322 13825 net.cpp:170] ip2\_ip2\_0\_split needs backward computation.
113. I0313 00:40:24.570338 13825 net.cpp:170] ip2 needs backward computation.
114. I0313 00:40:24.570349 13825 net.cpp:170] ip1 needs backward computation.
115. I0313 00:40:24.570359 13825 net.cpp:170] pool3 needs backward computation.
116. I0313 00:40:24.570372 13825 net.cpp:170] relu3 needs backward computation.
117. I0313 00:40:24.570384 13825 net.cpp:170] conv3 needs backward computation.
118. I0313 00:40:24.570396 13825 net.cpp:170] pool2 needs backward computation.
119. I0313 00:40:24.570406 13825 net.cpp:170] relu2 needs backward computation.
120. I0313 00:40:24.570420 13825 net.cpp:170] conv2 needs backward computation.
121. I0313 00:40:24.570432 13825 net.cpp:170] relu1 needs backward computation.
122. I0313 00:40:24.570442 13825 net.cpp:170] pool1 needs backward computation.
123. I0313 00:40:24.570456 13825 net.cpp:170] conv1 needs backward computation.
124. I0313 00:40:24.570471 13825 net.cpp:172] label\_cifar\_1\_split does not need backward computation.
125. I0313 00:40:24.570482 13825 net.cpp:172] cifar does not need backward computation.
126. I0313 00:40:24.570494 13825 net.cpp:208] This network produces output accuracy
127. I0313 00:40:24.570505 13825 net.cpp:208] This network produces output loss
128. I0313 00:40:24.570536 13825 net.cpp:467] Collecting Learning Rate and Weight Decay.
129. I0313 00:40:24.570549 13825 net.cpp:219] Network initialization done.
130. I0313 00:40:24.570554 13825 net.cpp:220] Memory required for data: 31987608
131. I0313 00:40:24.570590 13825 solver.cpp:41] Solver scaffolding done.
132. I0313 00:40:24.570595 13825 solver.cpp:160] Solving CIFAR10\_quick
133. I0313 00:40:24.570600 13825 solver.cpp:161] Learning Rate Policy: fixed

Top shape: 100 64 8 8 (409600)

额外补充：

如果调用CIFAR10的模型，会遇到要将binaryproto转成.npy格式的情况，需要调用python io模块里面的blobproto\_to\_array，能够生成npy，不过有的时候出现axes don't match array，

则需要修改这个函数， 可参见https://github.com/BVLC/caffe/issues/420

[学习笔记：AlexNet&Imagenet学习笔记](http://blog.csdn.net/lynnandwei/article/details/44411465)

分类： [Machine Learning](http://blog.csdn.net/lynnandwei/article/category/2864799)2015-03-18 18:20 554人阅读 [评论](http://blog.csdn.net/lynnandwei/article/details/44411465#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/lynnandwei/article/details/44411465#report)

ImageNet(http://www.image-net.org)是李菲菲组的图像库，和WordNet 可以结合使用 （毕业于Caltech；导师：Pietro Perona；主页：<http://vision.stanford.edu/~feifeili/>） 总共有十万的synset， 其中2010的数据表示，有图像的非空synset是21841，每一类大约1000张图片，图片总数：14197122。

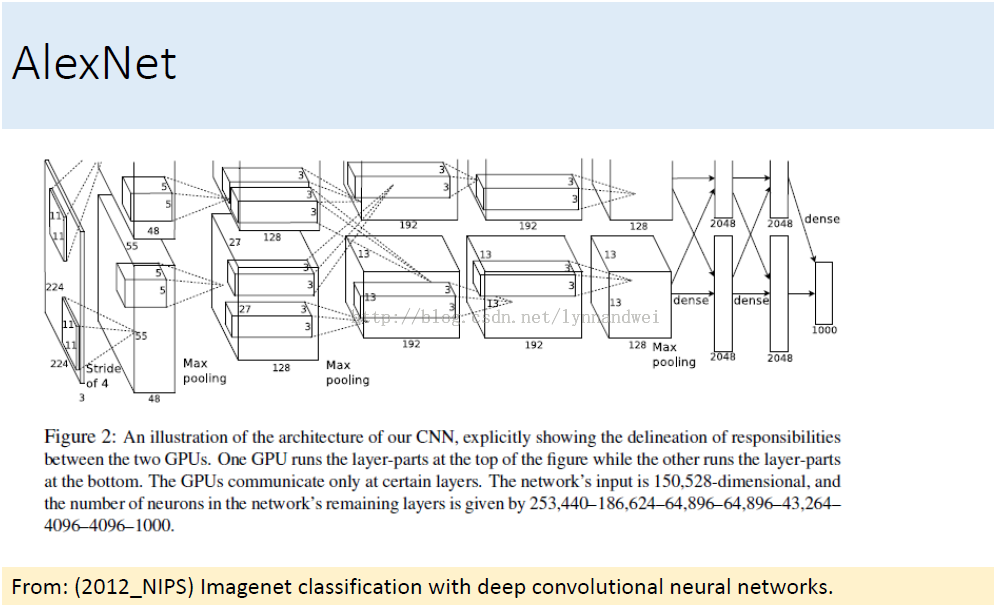
Caffe中训练ImageNet使用的是Alex Krizhevsky的方法（AlexNet），这个有60million参数的模型(简称为AlexNet)，在ILSVRC 2012中赢得了第一名,Top5错误率15.3%。

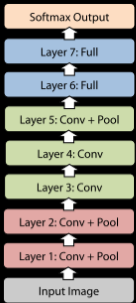
2013年，Clarifai通过cnn模型可视化技术调整网络架构，赢得了ILSVRC （http://www.clarifai.com/     看八卦 可以看[zhihu.com](http://zhuanlan.zhihu.com/cvprnet/19821292)   [Filestorm](http://www.zhihu.com/people/filestorm)：的一段闲聊： http://zhuanlan.zhihu.com/cvprnet/19821292）。

2014年，google也加入进来，它通过增加模型的层数（总共22层），并且利用multi-scale data training，取得第一名，Top5错误率 6.66%。Google的模型大大增加的网络的深度，并且去掉了最顶层的全连接层：因为全连接层（Fully Connected）几乎占据了CNN大概90%的参数，但是同时又可能带来过拟合（overfitting）的效果。它的模型比以前AlexNet的模型大大缩小，并且减轻了过拟合带来的副作用。Alex模型参数是60M，GoogLeNet只有7M。

2015年，百度用Deep Image,Top5错误率达到5.98% ,它基于GPU，利用36个服务节点开发了一个专为深度学习运算的supercompter(名叫Minwa，敏娲)。这台supercomputer具备TB级的host memory，超强的数据交换能力，训练了一个有100billion参数的巨大的深层神经网络。 （这一系统包含 36 个服务器节点，每一服务器节点配备了 2 颗六核英特尔至强 E5-2620 处理器。每个服务器包含 4 颗英伟达 Tesla K40m GPU，以及 1 个 FDR InfiniBand（速度为 56GB/S）。这带来了高性能、低延时的连接，以及对 RDMA 的支持。每一颗 GPU 的最高浮点运算性能为每秒 4.29 万亿次浮点运算，而每一颗 GPU 也配备了 12GB 的内存。整体来看，Minwa 内置了 6.9TB 的主内存、1.7TB 的设备内存，而理论上的最高性能约为 0.6 千万亿次浮点运算。凭借如此强大的系统，研究人员可以使用与其他深度学习项目不同，或者说更好的训练数据。因此，百度没有使用常见的 256x256 像素图片，而是使用了 512x512 像素图片，并且可以给这些图片添加各种特效，例如色彩调整、增加光晕，以及透镜扭曲等。这样做的目的是使系统学习更多尺寸更小的对象，并在各种环境下识别 对象。）

AlexNet整个模型比起之前我们看到的Cifar10 和LeNet模型相对来说复杂一些， 训练时间是在两台GTX 580 3GB GPUs上进行了5-6天的训练，其中用到了Hinton的改进方法（在全连接层加入ReLU+Dropout）, 对于这个模型，我们首先认定一个完整的卷积层可能包括一层convolution，一层Rectified Linear Units，一层max-pooling，一层normalization。则整个网络结构包括五层卷积层和三层全连接层，网络的最前端是输入图片的原始像素点，最后端是图片的分类结果。



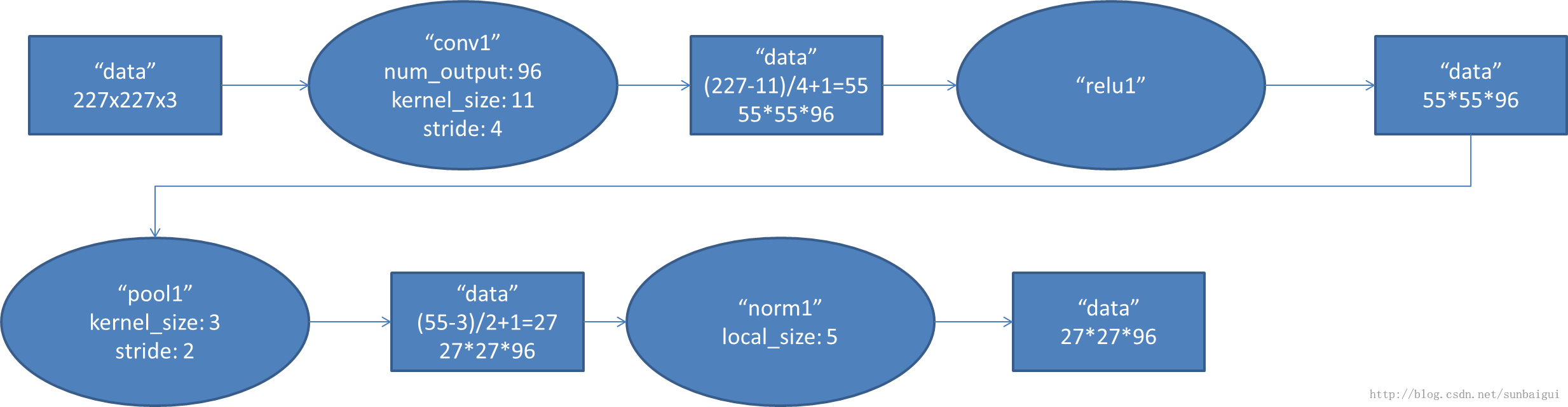
[](http://dataunion.org/wp-content/uploads/2015/02/convolution7.png)

对于每一层网络，具体的网络参数配置如下图所示。InputLayer就是输入图片层，每个输入图片都将被缩放成227\*227大小，分rgb三个颜色维度输入。Layer1~ Layer5是卷积层，以Layer1为例，卷积滤波器的大小是11\*11，卷积步幅为4，本层共有96个卷积滤波器，本层的输出则是96个55\*55大小的图片。在Layer1，卷积滤波后，还接有ReLUs操作和max-pooling操作。Layer6~ Layer8是全连接层，相当于在五层卷积层的基础上再加上一个三层的全连接神经网络分类器。以Layer6为例，本层的神经元个数为4096个。Layer8的神经元个数为1000个，相当于训练目标的1000个图片类别。

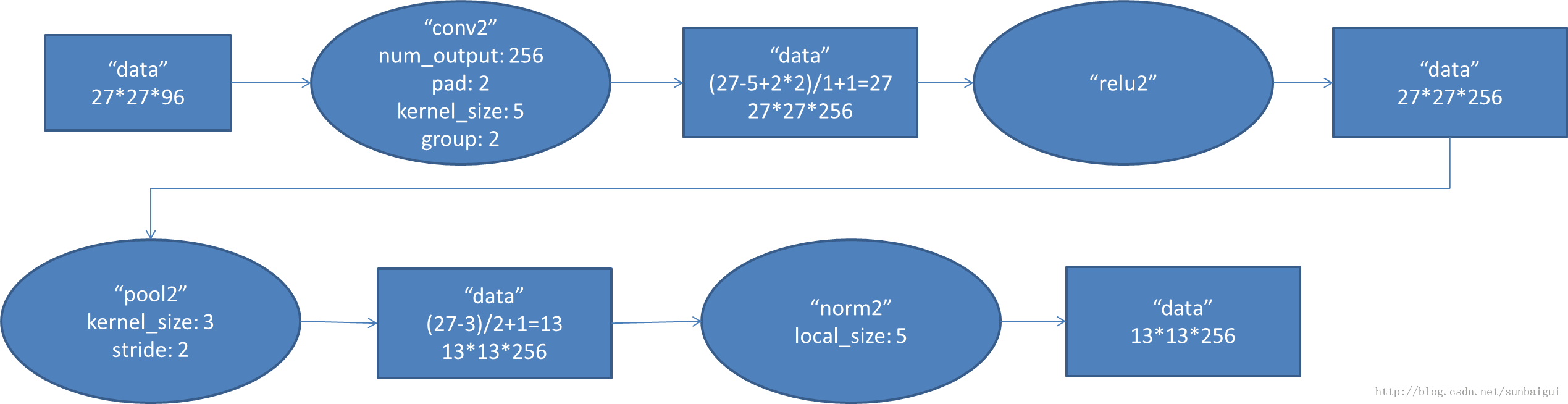
下面的图来自[**sunbaigui**](http://my.csdn.net/sunbaigui)的 博文  http://blog.csdn.net/sunbaigui/article/details/39938097

流程是一致的，不过对于每个值的计算过程 有点不一样。

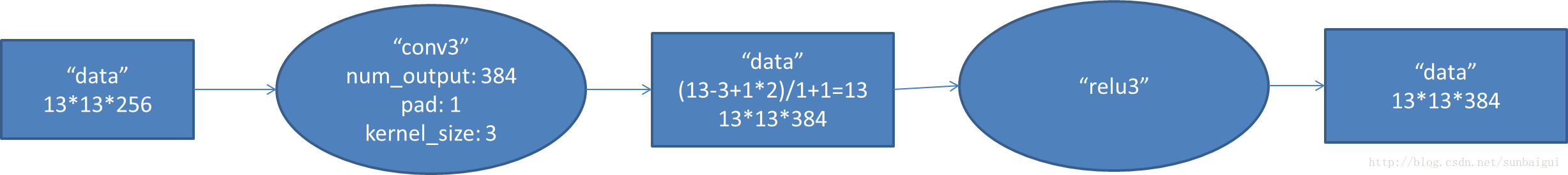
 conv1阶段DFD（data flow diagram）：



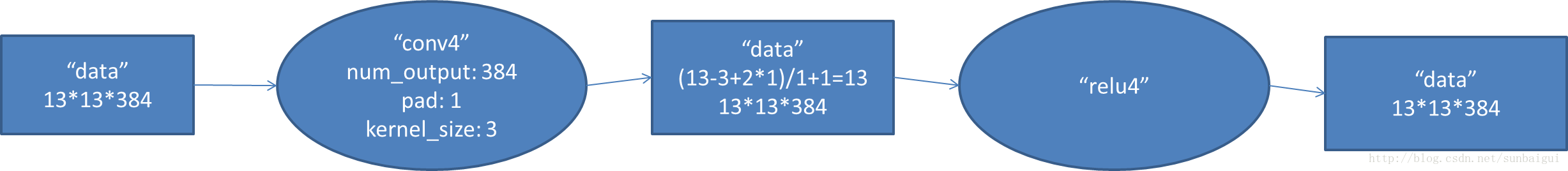
2. conv2阶段DFD（data flow diagram）：



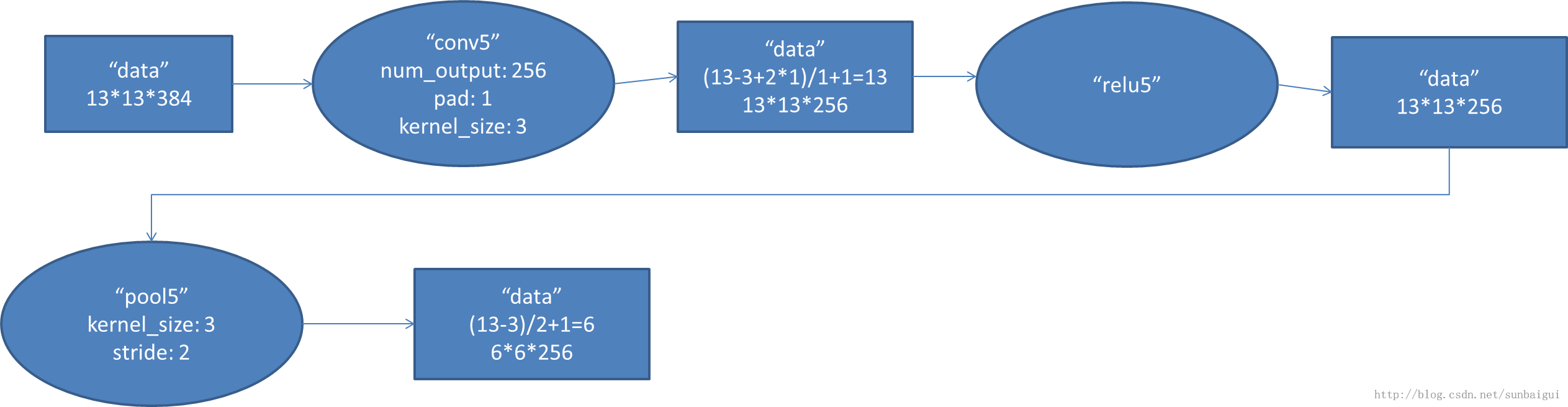
3. conv3阶段DFD（data flow diagram）：



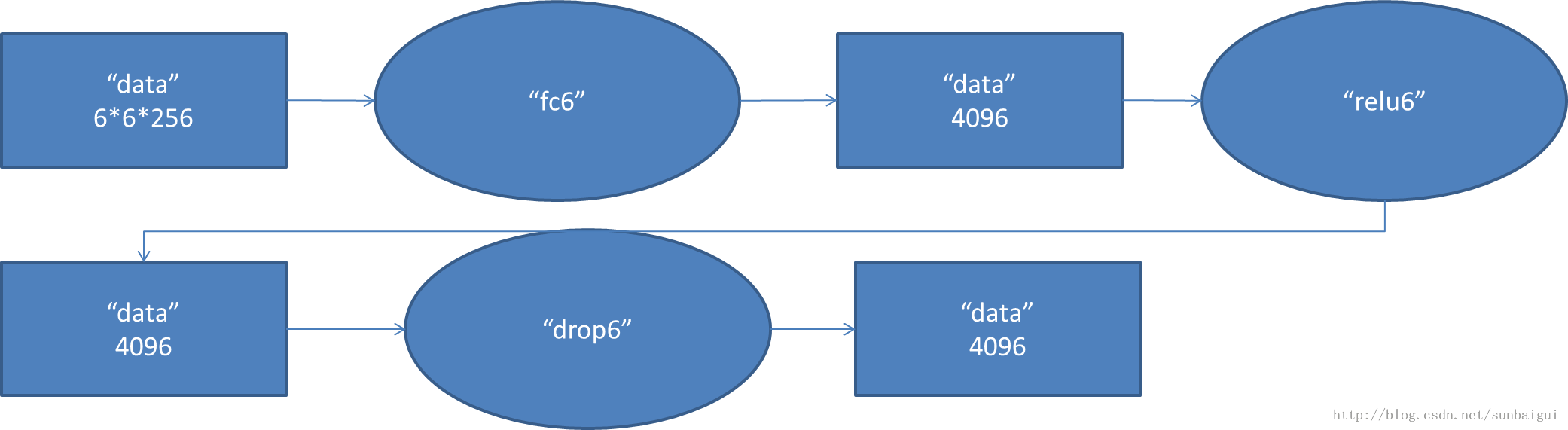
4. conv4阶段DFD（data flow diagram）：



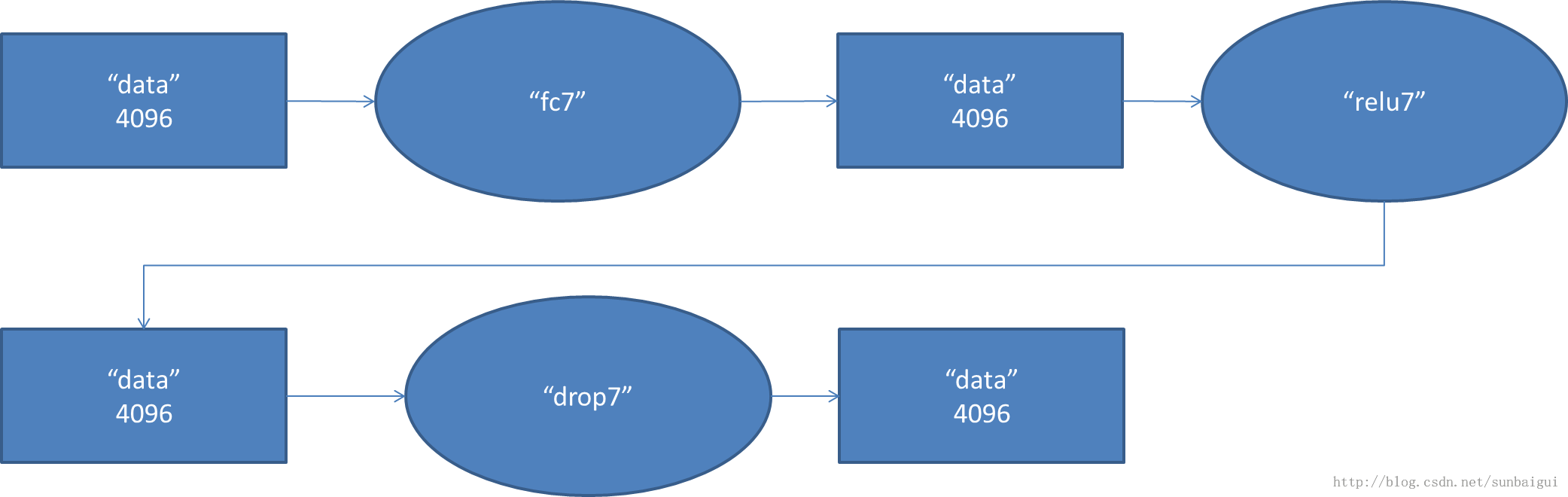
5. conv5阶段DFD（data flow diagram）：



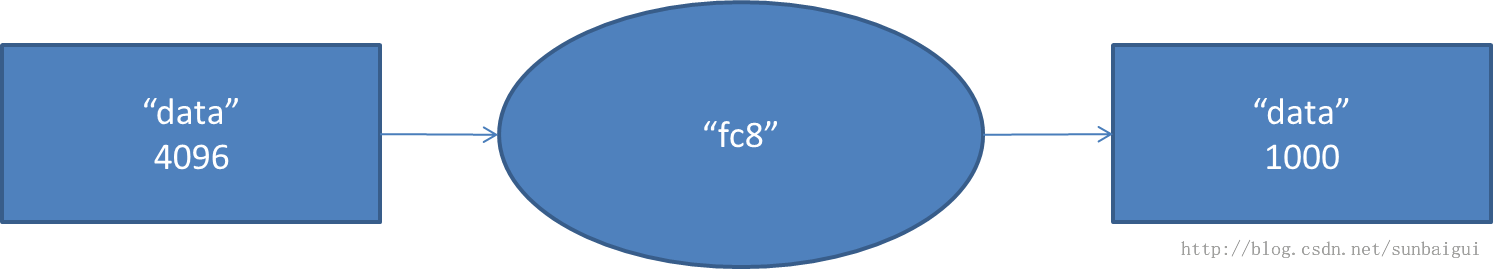
6. fc6阶段DFD（data flow diagram）：



7. fc7阶段DFD（data flow diagram）：



8. fc8阶段DFD（data flow diagram）：



然后我调用训练好的模型

MODEL\_FILE = '../models/bvlc\_reference\_caffenet/deploy.prototxt'  
PRETRAINED = '../models/bvlc\_reference\_caffenet/bvlc\_reference\_caffenet.caffemodel'

获取到预测结果，然后再标签里查找到对应的分类

imagenet\_label = caffe\_root + 'data/ilsvrc12/synset\_words.txt'

输入一张猫的图片



输出前五个分类

['n02123045 tabby, tabby cat' 'n02127052 lynx, catamount'  
 'n02124075 Egyptian cat' 'n07930864 cup' 'n02123159 tiger cat']  
[281 287 285 968 282]

**[html]** [view plaincopy](http://blog.csdn.net/lynnandwei/article/details/44411465)[在CODE上查看代码片](https://code.csdn.net/snippets/622644)

1. input: "data"
2. input\_dim: 10
3. input\_dim: 3
4. input\_dim: 227
5. input\_dim: 227
6. I0318 02:36:31.173322 15756 net.cpp:358] Input 0 -**>** data
7. I0318 02:36:31.173369 15756 net.cpp:56] Memory required for data: 0
8. I0318 02:36:31.173436 15756 net.cpp:67] Creating Layer conv1
9. I0318 02:36:31.173446 15756 net.cpp:394] conv1 **<-** data
10. I0318 02:36:31.173463 15756 net.cpp:356] conv1 -**>** conv1
11. I0318 02:36:31.173481 15756 net.cpp:96] Setting up conv1
12. I0318 02:36:31.173616 15756 net.cpp:103] Top shape: 10 96 55 55 (2904000)
13. I0318 02:36:31.173624 15756 net.cpp:113] Memory required for data: 11616000
14. I0318 02:36:31.173655 15756 net.cpp:67] Creating Layer relu1
15. I0318 02:36:31.173665 15756 net.cpp:394] relu1 **<-** conv1
16. I0318 02:36:31.173679 15756 net.cpp:345] relu1 -**>** conv1 (in-place)
17. I0318 02:36:31.173691 15756 net.cpp:96] Setting up relu1
18. I0318 02:36:31.173697 15756 net.cpp:103] Top shape: 10 96 55 55 (2904000)
19. I0318 02:36:31.173702 15756 net.cpp:113] Memory required for data: 23232000
20. I0318 02:36:31.173715 15756 net.cpp:67] Creating Layer pool1
21. I0318 02:36:31.173722 15756 net.cpp:394] pool1 **<-** conv1
22. I0318 02:36:31.173737 15756 net.cpp:356] pool1 -**>** pool1
23. I0318 02:36:31.173750 15756 net.cpp:96] Setting up pool1
24. I0318 02:36:31.173763 15756 net.cpp:103] Top shape: 10 96 27 27 (699840)
25. I0318 02:36:31.173768 15756 net.cpp:113] Memory required for data: 26031360
26. I0318 02:36:31.173780 15756 net.cpp:67] Creating Layer norm1
27. I0318 02:36:31.173787 15756 net.cpp:394] norm1 **<-** pool1
28. I0318 02:36:31.173800 15756 net.cpp:356] norm1 -**>** norm1
29. I0318 02:36:31.173811 15756 net.cpp:96] Setting up norm1
30. I0318 02:36:31.173820 15756 net.cpp:103] Top shape: 10 96 27 27 (699840)
31. I0318 02:36:31.173825 15756 net.cpp:113] Memory required for data: 28830720
32. I0318 02:36:31.173837 15756 net.cpp:67] Creating Layer conv2
33. I0318 02:36:31.173845 15756 net.cpp:394] conv2 **<-** norm1
34. I0318 02:36:31.173858 15756 net.cpp:356] conv2 -**>** conv2
35. I0318 02:36:31.173872 15756 net.cpp:96] Setting up conv2
36. I0318 02:36:31.174792 15756 net.cpp:103] Top shape: 10 256 27 27 (1866240)
37. I0318 02:36:31.174799 15756 net.cpp:113] Memory required for data: 36295680
38. I0318 02:36:31.174818 15756 net.cpp:67] Creating Layer relu2
39. I0318 02:36:31.174826 15756 net.cpp:394] relu2 **<-** conv2
40. I0318 02:36:31.174839 15756 net.cpp:345] relu2 -**>** conv2 (in-place)
41. I0318 02:36:31.174849 15756 net.cpp:96] Setting up relu2
42. I0318 02:36:31.174856 15756 net.cpp:103] Top shape: 10 256 27 27 (1866240)
43. I0318 02:36:31.174860 15756 net.cpp:113] Memory required for data: 43760640
44. I0318 02:36:31.174870 15756 net.cpp:67] Creating Layer pool2
45. I0318 02:36:31.174876 15756 net.cpp:394] pool2 **<-** conv2
46. I0318 02:36:31.174890 15756 net.cpp:356] pool2 -**>** pool2
47. I0318 02:36:31.174902 15756 net.cpp:96] Setting up pool2
48. I0318 02:36:31.174912 15756 net.cpp:103] Top shape: 10 256 13 13 (432640)
49. I0318 02:36:31.174917 15756 net.cpp:113] Memory required for data: 45491200
50. I0318 02:36:31.174927 15756 net.cpp:67] Creating Layer norm2
51. I0318 02:36:31.174933 15756 net.cpp:394] norm2 **<-** pool2
52. I0318 02:36:31.174947 15756 net.cpp:356] norm2 -**>** norm2
53. I0318 02:36:31.174957 15756 net.cpp:96] Setting up norm2
54. I0318 02:36:31.174967 15756 net.cpp:103] Top shape: 10 256 13 13 (432640)
55. I0318 02:36:31.174970 15756 net.cpp:113] Memory required for data: 47221760
56. I0318 02:36:31.174983 15756 net.cpp:67] Creating Layer conv3
57. I0318 02:36:31.174990 15756 net.cpp:394] conv3 **<-** norm2
58. I0318 02:36:31.175004 15756 net.cpp:356] conv3 -**>** conv3
59. I0318 02:36:31.175015 15756 net.cpp:96] Setting up conv3
60. I0318 02:36:31.177199 15756 net.cpp:103] Top shape: 10 384 13 13 (648960)
61. I0318 02:36:31.177208 15756 net.cpp:113] Memory required for data: 49817600
62. I0318 02:36:31.177230 15756 net.cpp:67] Creating Layer relu3
63. I0318 02:36:31.177238 15756 net.cpp:394] relu3 **<-** conv3
64. I0318 02:36:31.177252 15756 net.cpp:345] relu3 -**>** conv3 (in-place)
65. I0318 02:36:31.177263 15756 net.cpp:96] Setting up relu3
66. I0318 02:36:31.177268 15756 net.cpp:103] Top shape: 10 384 13 13 (648960)
67. I0318 02:36:31.177273 15756 net.cpp:113] Memory required for data: 52413440
68. I0318 02:36:31.177284 15756 net.cpp:67] Creating Layer conv4
69. I0318 02:36:31.177289 15756 net.cpp:394] conv4 **<-** conv3
70. I0318 02:36:31.177302 15756 net.cpp:356] conv4 -**>** conv4
71. I0318 02:36:31.177316 15756 net.cpp:96] Setting up conv4
72. I0318 02:36:31.179213 15756 net.cpp:103] Top shape: 10 384 13 13 (648960)
73. I0318 02:36:31.179220 15756 net.cpp:113] Memory required for data: 55009280
74. I0318 02:36:31.179237 15756 net.cpp:67] Creating Layer relu4
75. I0318 02:36:31.179244 15756 net.cpp:394] relu4 **<-** conv4
76. I0318 02:36:31.179258 15756 net.cpp:345] relu4 -**>** conv4 (in-place)
77. I0318 02:36:31.179268 15756 net.cpp:96] Setting up relu4
78. I0318 02:36:31.179275 15756 net.cpp:103] Top shape: 10 384 13 13 (648960)
79. I0318 02:36:31.179280 15756 net.cpp:113] Memory required for data: 57605120
80. I0318 02:36:31.179291 15756 net.cpp:67] Creating Layer conv5
81. I0318 02:36:31.179296 15756 net.cpp:394] conv5 **<-** conv4
82. I0318 02:36:31.179309 15756 net.cpp:356] conv5 -**>** conv5
83. I0318 02:36:31.179322 15756 net.cpp:96] Setting up conv5
84. I0318 02:36:31.180598 15756 net.cpp:103] Top shape: 10 256 13 13 (432640)
85. I0318 02:36:31.180606 15756 net.cpp:113] Memory required for data: 59335680
86. I0318 02:36:31.180629 15756 net.cpp:67] Creating Layer relu5
87. I0318 02:36:31.180636 15756 net.cpp:394] relu5 **<-** conv5
88. I0318 02:36:31.180649 15756 net.cpp:345] relu5 -**>** conv5 (in-place)
89. I0318 02:36:31.180660 15756 net.cpp:96] Setting up relu5
90. I0318 02:36:31.180665 15756 net.cpp:103] Top shape: 10 256 13 13 (432640)
91. I0318 02:36:31.180670 15756 net.cpp:113] Memory required for data: 61066240
92. I0318 02:36:31.180682 15756 net.cpp:67] Creating Layer pool5
93. I0318 02:36:31.180690 15756 net.cpp:394] pool5 **<-** conv5
94. I0318 02:36:31.180703 15756 net.cpp:356] pool5 -**>** pool5
95. I0318 02:36:31.180716 15756 net.cpp:96] Setting up pool5
96. I0318 02:36:31.180726 15756 net.cpp:103] Top shape: 10 256 6 6 (92160)
97. I0318 02:36:31.180730 15756 net.cpp:113] Memory required for data: 61434880
98. I0318 02:36:31.180742 15756 net.cpp:67] Creating Layer fc6
99. I0318 02:36:31.180748 15756 net.cpp:394] fc6 **<-** pool5
100. I0318 02:36:31.180760 15756 net.cpp:356] fc6 -**>** fc6
101. I0318 02:36:31.180773 15756 net.cpp:96] Setting up fc6
102. I0318 02:36:31.262177 15756 net.cpp:103] Top shape: 10 4096 1 1 (40960)
103. I0318 02:36:31.262205 15756 net.cpp:113] Memory required for data: 61598720
104. I0318 02:36:31.262254 15756 net.cpp:67] Creating Layer relu6
105. I0318 02:36:31.262269 15756 net.cpp:394] relu6 **<-** fc6
106. I0318 02:36:31.262296 15756 net.cpp:345] relu6 -**>** fc6 (in-place)
107. I0318 02:36:31.262312 15756 net.cpp:96] Setting up relu6
108. I0318 02:36:31.262320 15756 net.cpp:103] Top shape: 10 4096 1 1 (40960)
109. I0318 02:36:31.262325 15756 net.cpp:113] Memory required for data: 61762560
110. I0318 02:36:31.262341 15756 net.cpp:67] Creating Layer drop6
111. I0318 02:36:31.262346 15756 net.cpp:394] drop6 **<-** fc6
112. I0318 02:36:31.262358 15756 net.cpp:345] drop6 -**>** fc6 (in-place)
113. I0318 02:36:31.262369 15756 net.cpp:96] Setting up drop6
114. I0318 02:36:31.262378 15756 net.cpp:103] Top shape: 10 4096 1 1 (40960)
115. I0318 02:36:31.262382 15756 net.cpp:113] Memory required for data: 61926400
116. I0318 02:36:31.262408 15756 net.cpp:67] Creating Layer fc7
117. I0318 02:36:31.262416 15756 net.cpp:394] fc7 **<-** fc6
118. I0318 02:36:31.262430 15756 net.cpp:356] fc7 -**>** fc7
119. I0318 02:36:31.262445 15756 net.cpp:96] Setting up fc7
120. I0318 02:36:31.298831 15756 net.cpp:103] Top shape: 10 4096 1 1 (40960)
121. I0318 02:36:31.298859 15756 net.cpp:113] Memory required for data: 62090240
122. I0318 02:36:31.298895 15756 net.cpp:67] Creating Layer relu7
123. I0318 02:36:31.298908 15756 net.cpp:394] relu7 **<-** fc7
124. I0318 02:36:31.298933 15756 net.cpp:345] relu7 -**>** fc7 (in-place)
125. I0318 02:36:31.298949 15756 net.cpp:96] Setting up relu7
126. I0318 02:36:31.298955 15756 net.cpp:103] Top shape: 10 4096 1 1 (40960)
127. I0318 02:36:31.298960 15756 net.cpp:113] Memory required for data: 62254080
128. I0318 02:36:31.298974 15756 net.cpp:67] Creating Layer drop7
129. I0318 02:36:31.298981 15756 net.cpp:394] drop7 **<-** fc7
130. I0318 02:36:31.298993 15756 net.cpp:345] drop7 -**>** fc7 (in-place)
131. I0318 02:36:31.299003 15756 net.cpp:96] Setting up drop7
132. I0318 02:36:31.299011 15756 net.cpp:103] Top shape: 10 4096 1 1 (40960)
133. I0318 02:36:31.299016 15756 net.cpp:113] Memory required for data: 62417920
134. I0318 02:36:31.299028 15756 net.cpp:67] Creating Layer fc8
135. I0318 02:36:31.299034 15756 net.cpp:394] fc8 **<-** fc7
136. I0318 02:36:31.299046 15756 net.cpp:356] fc8 -**>** fc8
137. I0318 02:36:31.299060 15756 net.cpp:96] Setting up fc8
138. I0318 02:36:31.308233 15756 net.cpp:103] Top shape: 10 1000 1 1 (10000)
139. I0318 02:36:31.308260 15756 net.cpp:113] Memory required for data: 62457920
140. I0318 02:36:31.308295 15756 net.cpp:67] Creating Layer prob
141. I0318 02:36:31.308310 15756 net.cpp:394] prob **<-** fc8
142. I0318 02:36:31.308336 15756 net.cpp:356] prob -**>** prob
143. I0318 02:36:31.308353 15756 net.cpp:96] Setting up prob
144. I0318 02:36:31.308372 15756 net.cpp:103] Top shape: 10 1000 1 1 (10000)
145. I0318 02:36:31.308377 15756 net.cpp:113] Memory required for data: 62497920
146. I0318 02:36:31.308384 15756 net.cpp:172] prob does not need backward computation.
147. I0318 02:36:31.308404 15756 net.cpp:172] fc8 does not need backward computation.
148. I0318 02:36:31.308410 15756 net.cpp:172] drop7 does not need backward computation.
149. I0318 02:36:31.308415 15756 net.cpp:172] relu7 does not need backward computation.
150. I0318 02:36:31.308420 15756 net.cpp:172] fc7 does not need backward computation.
151. I0318 02:36:31.308425 15756 net.cpp:172] drop6 does not need backward computation.
152. I0318 02:36:31.308430 15756 net.cpp:172] relu6 does not need backward computation.
153. I0318 02:36:31.308435 15756 net.cpp:172] fc6 does not need backward computation.
154. I0318 02:36:31.308440 15756 net.cpp:172] pool5 does not need backward computation.
155. I0318 02:36:31.308445 15756 net.cpp:172] relu5 does not need backward computation.
156. I0318 02:36:31.308450 15756 net.cpp:172] conv5 does not need backward computation.
157. I0318 02:36:31.308454 15756 net.cpp:172] relu4 does not need backward computation.
158. I0318 02:36:31.308459 15756 net.cpp:172] conv4 does not need backward computation.
159. I0318 02:36:31.308465 15756 net.cpp:172] relu3 does not need backward computation.
160. I0318 02:36:31.308470 15756 net.cpp:172] conv3 does not need backward computation.
161. I0318 02:36:31.308473 15756 net.cpp:172] norm2 does not need backward computation.
162. I0318 02:36:31.308478 15756 net.cpp:172] pool2 does not need backward computation.
163. I0318 02:36:31.308483 15756 net.cpp:172] relu2 does not need backward computation.
164. I0318 02:36:31.308488 15756 net.cpp:172] conv2 does not need backward computation.
165. I0318 02:36:31.308493 15756 net.cpp:172] norm1 does not need backward computation.
166. I0318 02:36:31.308498 15756 net.cpp:172] pool1 does not need backward computation.
167. I0318 02:36:31.308502 15756 net.cpp:172] relu1 does not need backward computation.
168. I0318 02:36:31.308508 15756 net.cpp:172] conv1 does not need backward computation.
169. I0318 02:36:31.308512 15756 net.cpp:208] This network produces output prob
170. I0318 02:36:31.308542 15756 net.cpp:467] Collecting Learning Rate and Weight Decay.
171. I0318 02:36:31.308554 15756 net.cpp:219] Network initialization done.
172. I0318 02:36:31.308558 15756 net.cpp:220] Memory required for data: 62497920
173. E0318 02:36:31.683995 15756 upgrade\_proto.cpp:611] Attempting to upgrade input file specified using deprecated transformation parameters: ../models/bvlc\_reference\_caffenet/bvlc\_reference\_caffenet.caffemodel
174. I0318 02:36:31.684034 15756 upgrade\_proto.cpp:614] Successfully upgraded file specified using deprecated data transformation parameters.
175. E0318 02:36:31.684039 15756 upgrade\_proto.cpp:616] Note that future Caffe releases will only support transform\_param messages for transformation fields.
176. I0318 02:36:31.684048 15756 net.cpp:702] Ignoring source layer data
177. I0318 02:36:31.684052 15756 net.cpp:705] Copying source layer conv1
178. I0318 02:36:31.684367 15756 net.cpp:705] Copying source layer relu1
179. I0318 02:36:31.684375 15756 net.cpp:705] Copying source layer pool1
180. I0318 02:36:31.684380 15756 net.cpp:705] Copying source layer norm1
181. I0318 02:36:31.684383 15756 net.cpp:705] Copying source layer conv2
182. I0318 02:36:31.687021 15756 net.cpp:705] Copying source layer relu2
183. I0318 02:36:31.687028 15756 net.cpp:705] Copying source layer pool2
184. I0318 02:36:31.687033 15756 net.cpp:705] Copying source layer norm2
185. I0318 02:36:31.687038 15756 net.cpp:705] Copying source layer conv3
186. I0318 02:36:31.694574 15756 net.cpp:705] Copying source layer relu3
187. I0318 02:36:31.694586 15756 net.cpp:705] Copying source layer conv4
188. I0318 02:36:31.700258 15756 net.cpp:705] Copying source layer relu4
189. I0318 02:36:31.700266 15756 net.cpp:705] Copying source layer conv5
190. I0318 02:36:31.704053 15756 net.cpp:705] Copying source layer relu5
191. I0318 02:36:31.704062 15756 net.cpp:705] Copying source layer pool5
192. I0318 02:36:31.704067 15756 net.cpp:705] Copying source layer fc6
193. I0318 02:36:32.025676 15756 net.cpp:705] Copying source layer relu6
194. I0318 02:36:32.025704 15756 net.cpp:705] Copying source layer drop6
195. I0318 02:36:32.025709 15756 net.cpp:705] Copying source layer fc7
196. I0318 02:36:32.168704 15756 net.cpp:705] Copying source layer relu7
197. I0318 02:36:32.168730 15756 net.cpp:705] Copying source layer drop7
198. I0318 02:36:32.168735 15756 net.cpp:705] Copying source layer fc8
199. I0318 02:36:32.203642 15756 net.cpp:702] Ignoring source layer loss

['n02123045 tabby, tabby cat' 'n02127052 lynx, catamount'  
 'n02124075 Egyptian cat' 'n07930864 cup' 'n02123159 tiger cat']  
[281 287 285 968 282]

其他参考：

http://blog.csdn.net/sunbaigui/article/details/39938097

http://dataunion.org/10781.html

http://djt.qq.com/article/view/1245

[学习笔记：GoogLeNet](http://blog.csdn.net/lynnandwei/article/details/44458033)

分类： [Machine Learning](http://blog.csdn.net/lynnandwei/article/category/2864799)2015-03-19 15:02 926人阅读 [评论](http://blog.csdn.net/lynnandwei/article/details/44458033#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/lynnandwei/article/details/44458033#report)

GoogLeNet, 2014年ILSVRC挑战赛冠军，将Top5 的错误率降低到6.67%. 一个22层的深度网络，论文在http://arxiv.org/pdf/1409.4842v1.pdf，题目为：Going deeper with convolutions。（每次看这么简洁优雅的题目，就想吐槽国内写paper的 八股文题目）。GoogLeNet这个名字也是挺有意思的，为了像开山鼻祖的LeNet网络致敬，他们选择了这样的名字。

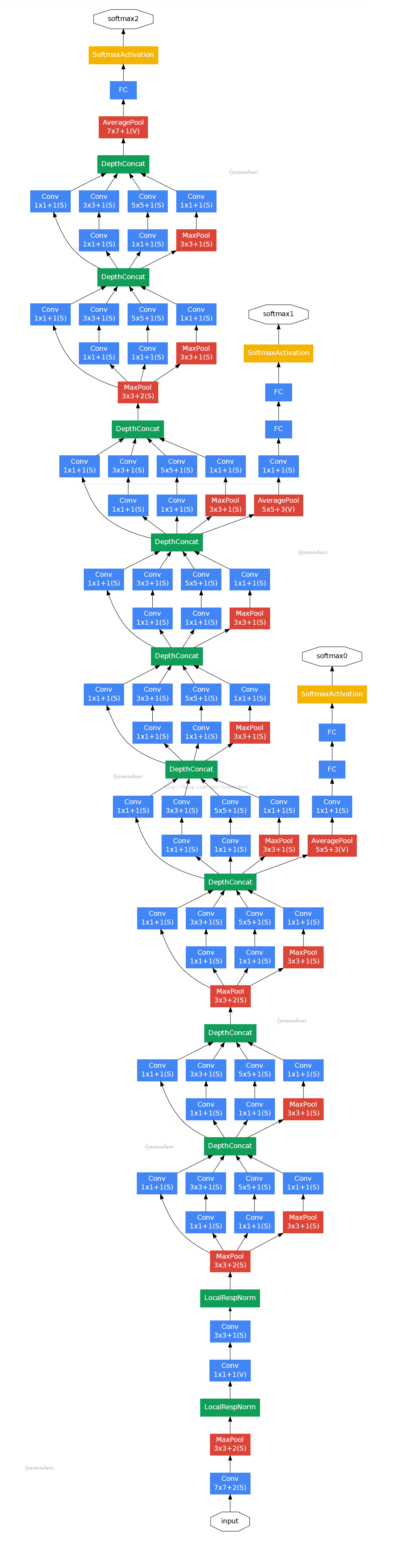
BVLC在caffe中给出了网络的实现：https://github.com/BVLC/caffe/tree/master/models/bvlc\_googlenet

模型下载地址：<http://dl.caffe.berkeleyvision.org/bvlc_googlenet.caffemodel>

从论文里整理了一张这个22个层次的模型的图出来（如果考虑pooling层是27层），先将模型跑了一遍结果，还是那只猫：

直观输出是如下，觉得挺准确：

[287 281 285 282 283]  
['n02127052 lynx, catamount' 'n02123045 tabby, tabby cat'  
 'n02124075 Egyptian cat' 'n02123159 tiger cat' 'n02123394 Persian cat']



caffe的实现和原来论文的模型是有不同的：

not training with the relighting data-augmentation;  
not training with the scale or aspect-ratio data-augmentation;  
uses "xavier" to initialize the weights instead of "gaussian";  
quick\_solver.prototxt uses a different learning rate decay policy than the original solver.prototxt, that allows a much faster training (60 epochs vs 250 epochs);  
The bundled model is the iteration 2,400,000 snapshot (60 epochs) using quick\_solver.prototxt

但是准确度还是达到了 a top-1 accuracy 68.7% (31.3% error) and a top-5 accuracy 88.9% (11.1% error)

我们来分析一下这个模型的层次关系:

原始数据，输入为224\*224\*3

第一层卷积层 conv1 ，pad是3，64个特征，7\*7 步长为2，输出特征为 112\*112\*64，然后进行relu，经过pool1（红色的max pool） 进行pooling 3\*3的核，步长为2， [（112 - 3+1）/2]+1 = 56  特征为56\*56\*64 ， 然后进行norm

第二层卷积层 conv2， pad是1，3\*3，192个特征，输出为56\*56\*192，然后进行relu，进行norm，经过pool2进行pooling，3\*3的核，步长为2 输出为28\*28\*192 然后进行split 分成四个支线

**[html]** [view plaincopy](http://blog.csdn.net/lynnandwei/article/details/44458033)[在CODE上查看代码片](https://code.csdn.net/snippets/623398)

1. Setting up pool2/3x3\_s2
2. I0319 23:50:37.405478  5765 net.cpp:103] Top shape: 10 192 28 28 (1505280)
3. I0319 23:50:37.405484  5765 net.cpp:113] Memory required for data: 174612480
4. I0319 23:50:37.405495  5765 net.cpp:67] Creating Layer pool2/3x3\_s2\_pool2/3x3\_s2\_0\_split
5. I0319 23:50:37.405503  5765 net.cpp:394] pool2/3x3\_s2\_pool2/3x3\_s2\_0\_split **<-** pool2/3x3\_s2
6. I0319 23:50:37.405515  5765 net.cpp:356] pool2/3x3\_s2\_pool2/3x3\_s2\_0\_split -**>** pool2/3x3\_s2\_pool2/3x3\_s2\_0\_split\_0
7. I0319 23:50:37.405531  5765 net.cpp:356] pool2/3x3\_s2\_pool2/3x3\_s2\_0\_split -**>** pool2/3x3\_s2\_pool2/3x3\_s2\_0\_split\_1
8. I0319 23:50:37.405545  5765 net.cpp:356] pool2/3x3\_s2\_pool2/3x3\_s2\_0\_split -**>** pool2/3x3\_s2\_pool2/3x3\_s2\_0\_split\_2
9. I0319 23:50:37.405557  5765 net.cpp:356] pool2/3x3\_s2\_pool2/3x3\_s2\_0\_split -**>** pool2/3x3\_s2\_pool2/3x3\_s2\_0\_split\_3
10. I0319 23:50:37.405567  5765 net.cpp:96] Setting up pool2/3x3\_s2\_pool2/3x3\_s2\_0\_split
11. I0319 23:50:37.405577  5765 net.cpp:103] Top shape: 10 192 28 28 (1505280)
12. I0319 23:50:37.405582  5765 net.cpp:103] Top shape: 10 192 28 28 (1505280)
13. I0319 23:50:37.405587  5765 net.cpp:103] Top shape: 10 192 28 28 (1505280)
14. I0319 23:50:37.405592  5765 net.cpp:103] Top shape: 10 192 28 28 (1505280)
15. I0319 23:50:37.405597  5765 net.cpp:113] Memory required for data: 198696960
16. I0319 23:50:37.405611  5765 net.cpp:67] Creating Layer inception\_3a/1x1^M

第三层开始时 inception module ，这个的思想受到使用不同尺度的Gabor过滤器来处理多尺度问题，inception module采用不同尺度的卷积核来处理问题。3a 包含 四个支线：

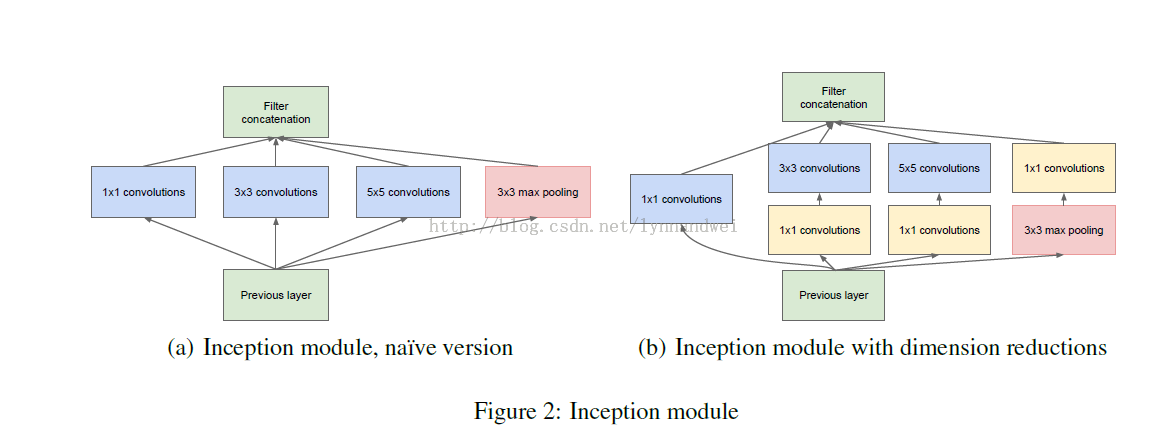
1： 64个1\*1的卷积核（之后进行RULE计算） 变成28\*28\*64

2： 96个1\*1的卷积核 作为3\*3卷积核之前的reduce，变成28\*28\*96， 进行relu计算后，再进行128个3\*3的卷积，pad为1， 28\*28\*128

3：16个1\*1的卷积核 作为5\*5卷积核之前的reduce，变成28\*28\*16， 进行relu计算后，再进行32个5\*5的卷积，pad为2，变成28\*28\*32

4：pool层，3\*3的核，pad为1，输出还是28\*28\*192，然后进行32个1\*1的卷积，变成28\*28\*32。

将四个结果进行连接，输出为28\*28\*256



然后将3a的结果又分成四条支线，开始建立3b的inception module

3b

1：128个1\*1的卷积核（之后进行RULE计算） 变成28\*28\*128

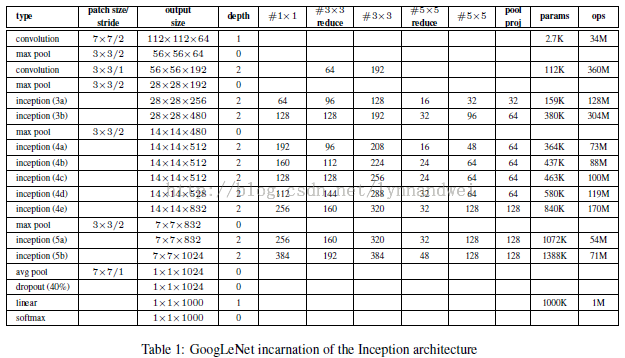
2：128个1\*1的卷积核 作为3\*3卷积核之前的reduce，变成28\*28\*128， 再进行192个3\*3的卷积，pad为1， 28\*28\*192，进行relu计算

3：32个1\*1的卷积核 作为5\*5卷积核之前的reduce，变成28\*28\*32， 进行relu计算后，再进行96个5\*5的卷积，pad为2，变成28\*28\*96

4：pool层，3\*3的核，pad为1，输出还是28\*28\*256，然后进行64个1\*1的卷积，变成28\*28\*64。

将四个结果进行连接，输出为28\*28\*480

同理依次推算，数据变化如下表：



一部分输出结果如下：

I0319 22:27:51.257917 5080 net.cpp:208] This network produces output prob

**[html]** [view plaincopy](http://blog.csdn.net/lynnandwei/article/details/44458033)[在CODE上查看代码片](https://code.csdn.net/snippets/623398)

1. I0319 22:27:51.258116  5080 net.cpp:467] Collecting Learning Rate and Weight Decay.
2. I0319 22:27:51.258162  5080 net.cpp:219] Network initialization done.
3. I0319 22:27:51.258167  5080 net.cpp:220] Memory required for data: 545512320
4. I0319 22:27:51.345417  5080 net.cpp:702] Ignoring source layer data
5. I0319 22:27:51.345443  5080 net.cpp:702] Ignoring source layer label\_data\_1\_split
6. I0319 22:27:51.345448  5080 net.cpp:705] Copying source layer conv1/7x7\_s2
7. I0319 22:27:51.345542  5080 net.cpp:705] Copying source layer conv1/relu\_7x7
8. I0319 22:27:51.345548  5080 net.cpp:705] Copying source layer pool1/3x3\_s2
9. I0319 22:27:51.345553  5080 net.cpp:705] Copying source layer pool1/norm1
10. I0319 22:27:51.345558  5080 net.cpp:705] Copying source layer conv2/3x3\_reduce
11. I0319 22:27:51.345600  5080 net.cpp:705] Copying source layer conv2/relu\_3x3\_reduce
12. I0319 22:27:51.345607  5080 net.cpp:705] Copying source layer conv2/3x3
13. I0319 22:27:51.346571  5080 net.cpp:705] Copying source layer conv2/relu\_3x3
14. I0319 22:27:51.346580  5080 net.cpp:705] Copying source layer conv2/norm2
15. I0319 22:27:51.346585  5080 net.cpp:705] Copying source layer pool2/3x3\_s2
16. I0319 22:27:51.346590  5080 net.cpp:705] Copying source layer pool2/3x3\_s2\_pool2/3x3\_s2\_0\_split
17. I0319 22:27:51.346595  5080 net.cpp:705] Copying source layer inception\_3a/1x1
18. I0319 22:27:51.346706  5080 net.cpp:705] Copying source layer inception\_3a/relu\_1x1
19. I0319 22:27:51.346712  5080 net.cpp:705] Copying source layer inception\_3a/3x3\_reduce
20. I0319 22:27:51.346879  5080 net.cpp:705] Copying source layer inception\_3a/relu\_3x3\_reduce
21. I0319 22:27:51.346885  5080 net.cpp:705] Copying source layer inception\_3a/3x3
22. I0319 22:27:51.347844  5080 net.cpp:705] Copying source layer inception\_3a/relu\_3x3
23. I0319 22:27:51.347851  5080 net.cpp:705] Copying source layer inception\_3a/5x5\_reduce
24. I0319 22:27:51.347885  5080 net.cpp:705] Copying source layer inception\_3a/relu\_5x5\_reduce
25. I0319 22:27:51.347892  5080 net.cpp:705] Copying source layer inception\_3a/5x5
26. I0319 22:27:51.348008  5080 net.cpp:705] Copying source layer inception\_3a/relu\_5x5
27. I0319 22:27:51.348014  5080 net.cpp:705] Copying source layer inception\_3a/pool
28. I0319 22:27:51.348019  5080 net.cpp:705] Copying source layer inception\_3a/pool\_proj
29. I0319 22:27:51.348080  5080 net.cpp:705] Copying source layer inception\_3a/relu\_pool\_proj
30. I0319 22:27:51.348085  5080 net.cpp:705] Copying source layer inception\_3a/output
31. I0319 22:27:51.348091  5080 net.cpp:705] Copying source layer inception\_3a/output\_inception\_3a/output\_0\_split
32. I0319 22:27:51.348096  5080 net.cpp:705] Copying source layer inception\_3b/1x1
33. I0319 22:27:51.348398  5080 net.cpp:705] Copying source layer inception\_3b/relu\_1x1
34. I0319 22:27:51.348405  5080 net.cpp:705] Copying source layer inception\_3b/3x3\_reduce
35. I0319 22:27:51.348700  5080 net.cpp:705] Copying source layer inception\_3b/relu\_3x3\_reduce
36. I0319 22:27:51.348707  5080 net.cpp:705] Copying source layer inception\_3b/3x3
37. I0319 22:27:51.350611  5080 net.cpp:705] Copying source layer inception\_3b/relu\_3x3
38. I0319 22:27:51.350620  5080 net.cpp:705] Copying source layer inception\_3b/5x5\_reduce
39. I0319 22:27:51.350699  5080 net.cpp:705] Copying source layer inception\_3b/relu\_5x5\_reduce
40. I0319 22:27:51.350705  5080 net.cpp:705] Copying source layer inception\_3b/5x5
41. I0319 22:27:51.351372  5080 net.cpp:705] Copying source layer inception\_3b/relu\_5x5
42. I0319 22:27:51.351378  5080 net.cpp:705] Copying source layer inception\_3b/pool
43. I0319 22:27:51.351384  5080 net.cpp:705] Copying source layer inception\_3b/pool\_proj
44. I0319 22:27:51.351546  5080 net.cpp:705] Copying source layer inception\_3b/relu\_pool\_proj
45. I0319 22:27:51.351552  5080 net.cpp:705] Copying source layer inception\_3b/output
46. I0319 22:27:51.351558  5080 net.cpp:705] Copying source layer pool3/3x3\_s2
47. I0319 22:27:51.351563  5080 net.cpp:705] Copying source layer pool3/3x3\_s2\_pool3/3x3\_s2\_0\_split
48. I0319 22:27:51.351569  5080 net.cpp:705] Copying source layer inception\_4a/1x1
49. I0319 22:27:51.352367  5080 net.cpp:705] Copying source layer inception\_4a/relu\_1x1
50. I0319 22:27:51.352375  5080 net.cpp:705] Copying source layer inception\_4a/3x3\_reduce
51. I0319 22:27:51.352782  5080 net.cpp:705] Copying source layer inception\_4a/relu\_3x3\_reduce
52. I0319 22:27:51.352789  5080 net.cpp:705] Copying source layer inception\_4a/3x3
53. I0319 22:27:51.354333  5080 net.cpp:705] Copying source layer inception\_4a/relu\_3x3
54. I0319 22:27:51.354341  5080 net.cpp:705] Copying source layer inception\_4a/5x5\_reduce
55. I0319 22:27:51.354420  5080 net.cpp:705] Copying source layer inception\_4a/relu\_5x5\_reduce
56. I0319 22:27:51.354429  5080 net.cpp:705] Copying source layer inception\_4a/5x5
57. I0319 22:27:51.354601  5080 net.cpp:705] Copying source layer inception\_4a/relu\_5x5
58. I0319 22:27:51.354609  5080 net.cpp:705] Copying source layer inception\_4a/pool
59. I0319 22:27:51.354614  5080 net.cpp:705] Copying source layer inception\_4a/pool\_proj
60. I0319 22:27:51.354887  5080 net.cpp:705] Copying source layer inception\_4a/relu\_pool\_proj
61. I0319 22:27:51.354894  5080 net.cpp:705] Copying source layer inception\_4a/output
62. I0319 22:27:51.354900  5080 net.cpp:705] Copying source layer inception\_4a/output\_inception\_4a/output\_0\_split
63. I0319 22:27:51.354910  5080 net.cpp:702] Ignoring source layer loss1/ave\_pool
64. I0319 22:27:51.354918  5080 net.cpp:702] Ignoring source layer loss1/conv
65. I0319 22:27:51.354923  5080 net.cpp:702] Ignoring source layer loss1/relu\_conv
66. I0319 22:27:51.354930  5080 net.cpp:702] Ignoring source layer loss1/fc
67. I0319 22:27:51.354936  5080 net.cpp:702] Ignoring source layer loss1/relu\_fc
68. I0319 22:27:51.354943  5080 net.cpp:702] Ignoring source layer loss1/drop\_fc
69. I0319 22:27:51.354950  5080 net.cpp:702] Ignoring source layer loss1/classifier
70. I0319 22:27:51.354956  5080 net.cpp:702] Ignoring source layer loss1/loss
71. I0319 22:27:51.354962  5080 net.cpp:705] Copying source layer inception\_4b/1x1
72. I0319 22:27:51.355681  5080 net.cpp:705] Copying source layer inception\_4b/relu\_1x1
73. I0319 22:27:51.355690  5080 net.cpp:705] Copying source layer inception\_4b/3x3\_reduce
74. I0319 22:27:51.356190  5080 net.cpp:705] Copying source layer inception\_4b/relu\_3x3\_reduce
75. I0319 22:27:51.356199  5080 net.cpp:705] Copying source layer inception\_4b/3x3
76. I0319 22:27:51.358134  5080 net.cpp:705] Copying source layer inception\_4b/relu\_3x3
77. I0319 22:27:51.358144  5080 net.cpp:705] Copying source layer inception\_4b/5x5\_reduce
78. I0319 22:27:51.358256  5080 net.cpp:705] Copying source layer inception\_4b/relu\_5x5\_reduce
79. I0319 22:27:51.358263  5080 net.cpp:705] Copying source layer inception\_4b/5x5
80. I0319 22:27:51.358608  5080 net.cpp:705] Copying source layer inception\_4b/relu\_5x5
81. I0319 22:27:51.358616  5080 net.cpp:705] Copying source layer inception\_4b/pool
82. I0319 22:27:51.358623  5080 net.cpp:705] Copying source layer inception\_4b/pool\_proj
83. I0319 22:27:51.358917  5080 net.cpp:705] Copying source layer inception\_4b/relu\_pool\_proj
84. I0319 22:27:51.358925  5080 net.cpp:705] Copying source layer inception\_4b/output
85. I0319 22:27:51.358932  5080 net.cpp:705] Copying source layer inception\_4b/output\_inception\_4b/output\_0\_split
86. I0319 22:27:51.358937  5080 net.cpp:705] Copying source layer inception\_4c/1x1
87. I0319 22:27:51.359519  5080 net.cpp:705] Copying source layer inception\_4c/relu\_1x1
88. I0319 22:27:51.359526  5080 net.cpp:705] Copying source layer inception\_4c/3x3\_reduce
89. I0319 22:27:51.360097  5080 net.cpp:705] Copying source layer inception\_4c/relu\_3x3\_reduce
90. I0319 22:27:51.360105  5080 net.cpp:705] Copying source layer inception\_4c/3x3
91. I0319 22:27:51.362634  5080 net.cpp:705] Copying source layer inception\_4c/relu\_3x3
92. I0319 22:27:51.362643  5080 net.cpp:705] Copying source layer inception\_4c/5x5\_reduce
93. I0319 22:27:51.362757  5080 net.cpp:705] Copying source layer inception\_4c/relu\_5x5\_reduce
94. I0319 22:27:51.362764  5080 net.cpp:705] Copying source layer inception\_4c/5x5
95. I0319 22:27:51.363106  5080 net.cpp:705] Copying source layer inception\_4c/relu\_5x5
96. I0319 22:27:51.363114  5080 net.cpp:705] Copying source layer inception\_4c/pool
97. I0319 22:27:51.363121  5080 net.cpp:705] Copying source layer inception\_4c/pool\_proj
98. I0319 22:27:51.363415  5080 net.cpp:705] Copying source layer inception\_4c/relu\_pool\_proj
99. I0319 22:27:51.363423  5080 net.cpp:705] Copying source layer inception\_4c/output
100. I0319 22:27:51.363430  5080 net.cpp:705] Copying source layer inception\_4c/output\_inception\_4c/output\_0\_split
101. I0319 22:27:51.363436  5080 net.cpp:705] Copying source layer inception\_4d/1x1
102. I0319 22:27:51.363937  5080 net.cpp:705] Copying source layer inception\_4d/relu\_1x1
103. I0319 22:27:51.363945  5080 net.cpp:705] Copying source layer inception\_4d/3x3\_reduce
104. I0319 22:27:51.364591  5080 net.cpp:705] Copying source layer inception\_4d/relu\_3x3\_reduce
105. I0319 22:27:51.364600  5080 net.cpp:705] Copying source layer inception\_4d/3x3
106. I0319 22:27:51.367797  5080 net.cpp:705] Copying source layer inception\_4d/relu\_3x3
107. I0319 22:27:51.367806  5080 net.cpp:705] Copying source layer inception\_4d/5x5\_reduce
108. I0319 22:27:51.367959  5080 net.cpp:705] Copying source layer inception\_4d/relu\_5x5\_reduce
109. I0319 22:27:51.367966  5080 net.cpp:705] Copying source layer inception\_4d/5x5
110. I0319 22:27:51.368420  5080 net.cpp:705] Copying source layer inception\_4d/relu\_5x5
111. I0319 22:27:51.368428  5080 net.cpp:705] Copying source layer inception\_4d/pool
112. I0319 22:27:51.368435  5080 net.cpp:705] Copying source layer inception\_4d/pool\_proj
113. I0319 22:27:51.368726  5080 net.cpp:705] Copying source layer inception\_4d/relu\_pool\_proj
114. I0319 22:27:51.368733  5080 net.cpp:705] Copying source layer inception\_4d/output
115. I0319 22:27:51.368739  5080 net.cpp:705] Copying source layer inception\_4d/output\_inception\_4d/output\_0\_split
116. I0319 22:27:51.368748  5080 net.cpp:702] Ignoring source layer loss2/ave\_pool
117. I0319 22:27:51.368755  5080 net.cpp:702] Ignoring source layer loss2/conv
118. I0319 22:27:51.368762  5080 net.cpp:702] Ignoring source layer loss2/relu\_conv
119. I0319 22:27:51.368768  5080 net.cpp:702] Ignoring source layer loss2/fc
120. I0319 22:27:51.368774  5080 net.cpp:702] Ignoring source layer loss2/relu\_fc
121. I0319 22:27:51.368780  5080 net.cpp:702] Ignoring source layer loss2/drop\_fc
122. I0319 22:27:51.368788  5080 net.cpp:702] Ignoring source layer loss2/classifier
123. I0319 22:27:51.368794  5080 net.cpp:702] Ignoring source layer loss2/loss
124. I0319 22:27:51.368800  5080 net.cpp:705] Copying source layer inception\_4e/1x1
125. I0319 22:27:51.369971  5080 net.cpp:705] Copying source layer inception\_4e/relu\_1x1
126. I0319 22:27:51.369981  5080 net.cpp:705] Copying source layer inception\_4e/3x3\_reduce
127. I0319 22:27:51.370717  5080 net.cpp:705] Copying source layer inception\_4e/relu\_3x3\_reduce
128. I0319 22:27:51.370725  5080 net.cpp:705] Copying source layer inception\_4e/3x3
129. I0319 22:27:51.374668  5080 net.cpp:705] Copying source layer inception\_4e/relu\_3x3
130. I0319 22:27:51.374678  5080 net.cpp:705] Copying source layer inception\_4e/5x5\_reduce
131. I0319 22:27:51.374831  5080 net.cpp:705] Copying source layer inception\_4e/relu\_5x5\_reduce
132. I0319 22:27:51.374840  5080 net.cpp:705] Copying source layer inception\_4e/5x5
133. I0319 22:27:51.375728  5080 net.cpp:705] Copying source layer inception\_4e/relu\_5x5
134. I0319 22:27:51.375737  5080 net.cpp:705] Copying source layer inception\_4e/pool
135. I0319 22:27:51.375744  5080 net.cpp:705] Copying source layer inception\_4e/pool\_proj
136. I0319 22:27:51.376332  5080 net.cpp:705] Copying source layer inception\_4e/relu\_pool\_proj
137. I0319 22:27:51.376340  5080 net.cpp:705] Copying source layer inception\_4e/output
138. I0319 22:27:51.376346  5080 net.cpp:705] Copying source layer pool4/3x3\_s2
139. I0319 22:27:51.376353  5080 net.cpp:705] Copying source layer pool4/3x3\_s2\_pool4/3x3\_s2\_0\_split
140. I0319 22:27:51.376359  5080 net.cpp:705] Copying source layer inception\_5a/1x1
141. I0319 22:27:51.378186  5080 net.cpp:705] Copying source layer inception\_5a/relu\_1x1
142. I0319 22:27:51.378196  5080 net.cpp:705] Copying source layer inception\_5a/3x3\_reduce
143. I0319 22:27:51.379343  5080 net.cpp:705] Copying source layer inception\_5a/relu\_3x3\_reduce
144. I0319 22:27:51.379353  5080 net.cpp:705] Copying source layer inception\_5a/3x3
145. I0319 22:27:51.383293  5080 net.cpp:705] Copying source layer inception\_5a/relu\_3x3
146. I0319 22:27:51.383303  5080 net.cpp:705] Copying source layer inception\_5a/5x5\_reduce
147. I0319 22:27:51.383548  5080 net.cpp:705] Copying source layer inception\_5a/relu\_5x5\_reduce
148. I0319 22:27:51.383558  5080 net.cpp:705] Copying source layer inception\_5a/5x5
149. I0319 22:27:51.384449  5080 net.cpp:705] Copying source layer inception\_5a/relu\_5x5
150. I0319 22:27:51.384459  5080 net.cpp:705] Copying source layer inception\_5a/pool
151. I0319 22:27:51.384465  5080 net.cpp:705] Copying source layer inception\_5a/pool\_proj
152. I0319 22:27:51.385397  5080 net.cpp:705] Copying source layer inception\_5a/relu\_pool\_proj
153. I0319 22:27:51.385406  5080 net.cpp:705] Copying source layer inception\_5a/output
154. I0319 22:27:51.385414  5080 net.cpp:705] Copying source layer inception\_5a/output\_inception\_5a/output\_0\_split
155. I0319 22:27:51.385421  5080 net.cpp:705] Copying source layer inception\_5b/1x1
156. I0319 22:27:51.388157  5080 net.cpp:705] Copying source layer inception\_5b/relu\_1x1
157. I0319 22:27:51.388166  5080 net.cpp:705] Copying source layer inception\_5b/3x3\_reduce
158. I0319 22:27:51.389539  5080 net.cpp:705] Copying source layer inception\_5b/relu\_3x3\_reduce
159. I0319 22:27:51.389549  5080 net.cpp:705] Copying source layer inception\_5b/3x3
160. I0319 22:27:51.395212  5080 net.cpp:705] Copying source layer inception\_5b/relu\_3x3
161. I0319 22:27:51.395225  5080 net.cpp:705] Copying source layer inception\_5b/5x5\_reduce
162. I0319 22:27:51.395584  5080 net.cpp:705] Copying source layer inception\_5b/relu\_5x5\_reduce
163. I0319 22:27:51.395594  5080 net.cpp:705] Copying source layer inception\_5b/5x5
164. I0319 22:27:51.396921  5080 net.cpp:705] Copying source layer inception\_5b/relu\_5x5
165. I0319 22:27:51.396931  5080 net.cpp:705] Copying source layer inception\_5b/pool
166. I0319 22:27:51.396939  5080 net.cpp:705] Copying source layer inception\_5b/pool\_proj
167. I0319 22:27:51.397862  5080 net.cpp:705] Copying source layer inception\_5b/relu\_pool\_proj
168. I0319 22:27:51.397871  5080 net.cpp:705] Copying source layer inception\_5b/output
169. I0319 22:27:51.397879  5080 net.cpp:705] Copying source layer pool5/7x7\_s1
170. I0319 22:27:51.397886  5080 net.cpp:705] Copying source layer pool5/drop\_7x7\_s1
171. I0319 22:27:51.397893  5080 net.cpp:705] Copying source layer loss3/classifier
172. I0319 22:27:51.406652  5080 net.cpp:702] Ignoring source layer loss3/loss3

材料集合：

http://deeplearning.net/2014/09/19/googles-entry-to-imagenet-2014-challenge/

[1] Imagenet 2014 LSVRC results, [**http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/,**](http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/,)Last retrieved on: 19-09-2014.

[2] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Going Deeper with Convolutions, Arxiv Link: [**http://arxiv.org/abs/1409.4842**](http://arxiv.org/abs/1409.4842).

[3] GoogLeNet presentation, [**http://image-net.org/challenges/LSVRC/2014/slides/GoogLeNet.pptx**](http://image-net.org/challenges/LSVRC/2014/slides/GoogLeNet.pptx), Last retrieved on: 19-09.2014..

[4] What I learned from competing against a convnet on imagenet, [**http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet**](http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet)/, Last retrieved on: 19-09-2014.

[5] Girshick, Ross, et al. “Rich feature hierarchies for accurate object detection and semantic segmentation.” *arXiv preprint arXiv:1311.2524* (2013).

# [学习笔记：自训练Cifar10网络数据结果](http://blog.csdn.net/lynnandwei/article/details/45503511)

分类： [Machine Learning](http://blog.csdn.net/lynnandwei/article/category/2864799)2015-05-05 17:51 150人阅读 [评论](http://blog.csdn.net/lynnandwei/article/details/45503511#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/lynnandwei/article/details/45503511#report)

看了那么多深度网络的结构，于是上个月就开始自己拿数据来训练。

训练的数据需要尽量多一点。

首先拿101\_ObjectCategories的数据来试一下。101\_ObjectCategories里面有的类别数据量太少，于是先选择两类，airplanes 和Motorbikes  每一类有800张左右的图片。

训练图片还是32\*32. 模型基本没做大的修改。直接训练两类的结果 迭代到500次，准确率就达到99%，最后进行了4000次迭代。

I0408 23:46:38.619588 11493 net.cpp:56] Memory required for data: 0  
I0408 23:46:38.619616 11493 net.cpp:67] Creating Layer cifar  
I0408 23:46:38.619624 11493 net.cpp:356] cifar -> data  
I0408 23:46:38.619638 11493 net.cpp:356] cifar -> label  
I0408 23:46:38.619648 11493 net.cpp:96] Setting up cifar  
I0408 23:46:38.619704 11493 data\_layer.cpp:68] Opening lmdb data/101small/test32/101\_val\_lmdb  
I0408 23:46:38.619730 11493 data\_layer.cpp:128] output data size: 100,3,32,32  
I0408 23:46:38.619743 11493 base\_data\_layer.cpp:36] Loading mean file fromdata/101small/test32/mean.binaryproto  
I0408 23:46:38.620277 11493 base\_data\_layer.cpp:64] Initializing prefetch  
I0408 23:46:38.620301 11493 base\_data\_layer.cpp:66] Prefetch initialized.  
I0408 23:46:38.620309 11493 net.cpp:103] Top shape: 100 3 32 32 (307200)  
I0408 23:46:38.620316 11493 net.cpp:103] Top shape: 100 1 1 1 (100)  
I0408 23:46:38.620319 11493 net.cpp:113] Memory required for data: 1229200  
I0408 23:46:38.620331 11493 net.cpp:67] Creating Layer label\_cifar\_1\_split  
I0408 23:46:38.620337 11493 net.cpp:394] label\_cifar\_1\_split <- label  
I0408 23:46:38.620347 11493 net.cpp:356] label\_cifar\_1\_split -> label\_cifar\_1\_split\_0  
I0408 23:46:38.620357 11493 net.cpp:356] label\_cifar\_1\_split -> label\_cifar\_1\_split\_1  
I0408 23:46:38.620367 11493 net.cpp:96] Setting up label\_cifar\_1\_split  
I0408 23:46:38.620374 11493 net.cpp:103] Top shape: 100 1 1 1 (100)  
I0408 23:46:38.620379 11493 net.cpp:103] Top shape: 100 1 1 1 (100)  
I0408 23:46:38.620384 11493 net.cpp:113] Memory required for data: 1230000  
I0408 23:46:38.620395 11493 net.cpp:67] Creating Layer conv1  
I0408 23:46:38.620403 11493 net.cpp:394] conv1 <- data  
I0408 23:46:38.620412 11493 net.cpp:356] conv1 -> conv1  
I0408 23:46:38.620422 11493 net.cpp:96] Setting up conv1  
I0408 23:46:38.620764 11493 net.cpp:103] Top shape: 100 32 32 32 (3276800)  
I0408 23:46:38.620771 11493 net.cpp:113] Memory required for data: 14337200  
I0408 23:46:38.620786 11493 net.cpp:67] Creating Layer pool1  
I0408 23:46:38.620793 11493 net.cpp:394] pool1 <- conv1  
I0408 23:46:38.620803 11493 net.cpp:356] pool1 -> pool1  
I0408 23:46:38.620812 11493 net.cpp:96] Setting up pool1  
I0408 23:46:38.620820 11493 net.cpp:103] Top shape: 100 32 16 16 (819200)  
I0408 23:46:38.620826 11493 net.cpp:113] Memory required for data: 17614000  
I0408 23:46:38.620834 11493 net.cpp:67] Creating Layer relu1  
I0408 23:46:38.620839 11493 net.cpp:394] relu1 <- pool1  
I0408 23:46:38.620851 11493 net.cpp:345] relu1 -> pool1 (in-place)  
I0408 23:46:38.620867 11493 net.cpp:96] Setting up relu1  
I0408 23:46:38.620872 11493 net.cpp:103] Top shape: 100 32 16 16 (819200)  
I0408 23:46:38.620877 11493 net.cpp:113] Memory required for data: 20890800  
I0408 23:46:38.620887 11493 net.cpp:67] Creating Layer conv2  
I0408 23:46:38.620893 11493 net.cpp:394] conv2 <- pool1  
I0408 23:46:38.620903 11493 net.cpp:356] conv2 -> conv2  
I0408 23:46:38.620913 11493 net.cpp:96] Setting up conv2  
I0408 23:46:38.624178 11493 net.cpp:103] Top shape: 100 32 16 16 (819200)  
I0408 23:46:38.624187 11493 net.cpp:113] Memory required for data: 24167600  
I0408 23:46:38.624202 11493 net.cpp:67] Creating Layer relu2  
I0408 23:46:38.624215 11493 net.cpp:394] relu2 <- conv2  
I0408 23:46:38.624227 11493 net.cpp:345] relu2 -> conv2 (in-place)  
I0408 23:46:38.624235 11493 net.cpp:96] Setting up relu2  
I0408 23:46:38.624241 11493 net.cpp:103] Top shape: 100 32 16 16 (819200)  
I0408 23:46:38.624245 11493 net.cpp:113] Memory required for data: 27444400  
I0408 23:46:38.624253 11493 net.cpp:67] Creating Layer pool2  
I0408 23:46:38.624259 11493 net.cpp:394] pool2 <- conv2  
I0408 23:46:38.624269 11493 net.cpp:356] pool2 -> pool2  
I0408 23:46:38.624277 11493 net.cpp:96] Setting up pool2  
I0408 23:46:38.624284 11493 net.cpp:103] Top shape: 100 32 8 8 (204800)  
I0408 23:46:38.624289 11493 net.cpp:113] Memory required for data: 28263600  
I0408 23:46:38.624302 11493 net.cpp:67] Creating Layer conv3  
I0408 23:46:38.624308 11493 net.cpp:394] conv3 <- pool2  
I0408 23:46:38.624318 11493 net.cpp:356] conv3 -> conv3  
I0408 23:46:38.624330 11493 net.cpp:96] Setting up conv3  
I0408 23:46:38.630815 11493 net.cpp:103] Top shape: 100 64 8 8 (409600)  
I0408 23:46:38.630823 11493 net.cpp:113] Memory required for data: 29902000  
I0408 23:46:38.630838 11493 net.cpp:67] Creating Layer relu3  
I0408 23:46:38.630846 11493 net.cpp:394] relu3 <- conv3  
I0408 23:46:38.630856 11493 net.cpp:345] relu3 -> conv3 (in-place)  
I0408 23:46:38.630863 11493 net.cpp:96] Setting up relu3  
I0408 23:46:38.630869 11493 net.cpp:103] Top shape: 100 64 8 8 (409600)  
I0408 23:46:38.630874 11493 net.cpp:113] Memory required for data: 31540400  
I0408 23:46:38.630882 11493 net.cpp:67] Creating Layer pool3  
I0408 23:46:38.630887 11493 net.cpp:394] pool3 <- conv3  
I0408 23:46:38.630897 11493 net.cpp:356] pool3 -> pool3  
I0408 23:46:38.630905 11493 net.cpp:96] Setting up pool3  
I0408 23:46:38.630913 11493 net.cpp:103] Top shape: 100 64 4 4 (102400)  
I0408 23:46:38.630918 11493 net.cpp:113] Memory required for data: 31950000  
I0408 23:46:38.630926 11493 net.cpp:67] Creating Layer ip1  
I0408 23:46:38.630933 11493 net.cpp:394] ip1 <- pool3  
I0408 23:46:38.630942 11493 net.cpp:356] ip1 -> ip1  
I0408 23:46:38.630951 11493 net.cpp:96] Setting up ip1  
I0408 23:46:38.639205 11493 net.cpp:103] Top shape: 100 64 1 1 (6400)  
I0408 23:46:38.639214 11493 net.cpp:113] Memory required for data: 31975600  
I0408 23:46:38.639225 11493 net.cpp:67] Creating Layer ip2  
I0408 23:46:38.639232 11493 net.cpp:394] ip2 <- ip1  
I0408 23:46:38.639245 11493 net.cpp:356] ip2 -> ip2  
I0408 23:46:38.639255 11493 net.cpp:96] Setting up ip2  
I0408 23:46:38.639284 11493 net.cpp:103] Top shape: 100 2 1 1 (200)  
I0408 23:46:38.639291 11493 net.cpp:113] Memory required for data: 31976400  
I0408 23:46:38.639303 11493 net.cpp:67] Creating Layer ip2\_ip2\_0\_split  
I0408 23:46:38.639309 11493 net.cpp:394] ip2\_ip2\_0\_split <- ip2  
I0408 23:46:38.639318 11493 net.cpp:356] ip2\_ip2\_0\_split -> ip2\_ip2\_0\_split\_0  
I0408 23:46:38.639330 11493 net.cpp:356] ip2\_ip2\_0\_split -> ip2\_ip2\_0\_split\_1  
I0408 23:46:38.639339 11493 net.cpp:96] Setting up ip2\_ip2\_0\_split  
I0408 23:46:38.639346 11493 net.cpp:103] Top shape: 100 2 1 1 (200)  
I0408 23:46:38.639351 11493 net.cpp:103] Top shape: 100 2 1 1 (200)  
I0408 23:46:38.639355 11493 net.cpp:113] Memory required for data: 31978000  
I0408 23:46:38.639364 11493 net.cpp:67] Creating Layer accuracy  
I0408 23:46:38.639369 11493 net.cpp:394] accuracy <- ip2\_ip2\_0\_split\_0  
I0408 23:46:38.639376 11493 net.cpp:394] accuracy <- label\_cifar\_1\_split\_0  
I0408 23:46:38.639385 11493 net.cpp:356] accuracy -> accuracy  
I0408 23:46:38.639396 11493 net.cpp:96] Setting up accuracy  
I0408 23:46:38.639405 11493 net.cpp:103] Top shape: 1 1 1 1 (1)  
I0408 23:46:38.639410 11493 net.cpp:113] Memory required for data: 31978004  
I0408 23:46:38.639417 11493 net.cpp:67] Creating Layer loss  
I0408 23:46:38.639423 11493 net.cpp:394] loss <- ip2\_ip2\_0\_split\_1  
I0408 23:46:38.639431 11493 net.cpp:394] loss <- label\_cifar\_1\_split\_1  
I0408 23:46:38.639438 11493 net.cpp:356] loss -> loss  
I0408 23:46:38.639447 11493 net.cpp:96] Setting up loss  
I0408 23:46:38.639458 11493 net.cpp:103] Top shape: 1 1 1 1 (1)  
I0408 23:46:38.639463 11493 net.cpp:109]     with loss weight 1  
I0408 23:46:38.639469 11493 net.cpp:113] Memory required for data: 31978008  
I0408 23:46:38.639482 11493 net.cpp:170] loss needs backward computation.  
I0408 23:46:38.639488 11493 net.cpp:172] accuracy does not need backward computation.  
I0408 23:46:38.639493 11493 net.cpp:170] ip2\_ip2\_0\_split needs backward computation.  
I0408 23:46:38.639498 11493 net.cpp:170] ip2 needs backward computation.  
I0408 23:46:38.639503 11493 net.cpp:170] ip1 needs backward computation.  
I0408 23:46:38.639508 11493 net.cpp:170] pool3 needs backward computation.  
I0408 23:46:38.639513 11493 net.cpp:170] relu3 needs backward computation.  
I0408 23:46:38.639518 11493 net.cpp:170] conv3 needs backward computation.  
I0408 23:46:38.639523 11493 net.cpp:170] pool2 needs backward computation.  
I0408 23:46:38.639528 11493 net.cpp:170] relu2 needs backward computation.  
I0408 23:46:38.639533 11493 net.cpp:170] conv2 needs backward computation.  
I0408 23:46:38.639539 11493 net.cpp:170] relu1 needs backward computation.  
I0408 23:46:38.639544 11493 net.cpp:170] pool1 needs backward computation.  
I0408 23:46:38.639549 11493 net.cpp:170] conv1 needs backward computation.  
I0408 23:46:38.639554 11493 net.cpp:172] label\_cifar\_1\_split does not need backward computation.  
I0408 23:46:38.639559 11493 net.cpp:172] cifar does not need backward computation.  
I0408 23:46:38.639564 11493 net.cpp:208] This network produces output accuracy  
I0408 23:46:38.639569 11493 net.cpp:208] This network produces output loss  
I0408 23:46:38.639590 11493 net.cpp:467] Collecting Learning Rate and Weight Decay.  
I0408 23:46:38.639598 11493 net.cpp:219] Network initialization done.  
I0408 23:46:38.639603 11493 net.cpp:220] Memory required for data: 31978008  
I0408 23:46:38.639637 11493 solver.cpp:41] Solver scaffolding done.  
I0408 23:46:38.639643 11493 solver.cpp:160] Solving CIFAR10\_quick  
I0408 23:46:38.639648 11493 solver.cpp:161] Learning Rate Policy: fixed  
I0408 23:46:38.639696 11493 solver.cpp:264] Iteration 0, Testing net (#0)  
I0408 23:46:38.639713 11493 net.cpp:652] Copying source layer cifar  
I0408 23:46:38.639721 11493 net.cpp:652] Copying source layer conv1  
I0408 23:46:38.639736 11493 net.cpp:652] Copying source layer pool1  
I0408 23:46:38.639749 11493 net.cpp:652] Copying source layer relu1  
I0408 23:46:38.639757 11493 net.cpp:652] Copying source layer conv2  
I0408 23:46:38.639770 11493 net.cpp:652] Copying source layer relu2  
I0408 23:46:38.639783 11493 net.cpp:652] Copying source layer pool2  
I0408 23:46:38.639796 11493 net.cpp:652] Copying source layer conv3  
I0408 23:46:38.639822 11493 net.cpp:652] Copying source layer relu3  
I0408 23:46:38.639832 11493 net.cpp:652] Copying source layer pool3  
I0408 23:46:38.639838 11493 net.cpp:652] Copying source layer ip1  
I0408 23:46:38.639858 11493 net.cpp:652] Copying source layer ip2  
I0408 23:46:38.639864 11493 net.cpp:652] Copying source layer loss  
I0408 23:46:38.641947 11496 data\_layer.cpp:195] Restarting data prefetching from start.  
I0408 23:47:34.499454 11493 solver.cpp:315]     Test net output #0: accuracy = 0.505  
I0408 23:47:34.499495 11493 solver.cpp:315]     Test net output #1: loss = 0.693705 (\* 1 = 0.693705 loss)  
I0408 23:47:35.913596 11493 solver.cpp:209] Iteration 0, loss = 0.695267  
I0408 23:47:35.913638 11493 solver.cpp:224]     Train net output #0: loss = 0.695267 (\* 1 = 0.695267 loss)  
I0408 23:47:35.913648 11493 solver.cpp:445] Iteration 0, lr = 0.001  
I0408 23:47:51.280112 11612 data\_layer.cpp:195] Restarting data prefetching from start.  
I0408 23:49:55.618027 11493 solver.cpp:209] Iteration 100, loss = 0.0746845  
I0408 23:49:55.618070 11493 solver.cpp:224]     Train net output #0: loss = 0.0746845 (\* 1 = 0.0746845 loss)  
I0408 23:49:55.618079 11493 solver.cpp:445] Iteration 100, lr = 0.001  
I0408 23:50:08.198837 11752 data\_layer.cpp:195] Restarting data prefetching from start.  
I0408 23:52:15.567517 11493 solver.cpp:209] Iteration 200, loss = 0.0127414  
I0408 23:52:15.567559 11493 solver.cpp:224]     Train net output #0: loss = 0.0127414 (\* 1 = 0.0127414 loss)  
I0408 23:52:15.567569 11493 solver.cpp:445] Iteration 200, lr = 0.001  
I0408 23:52:25.373136 11884 data\_layer.cpp:195] Restarting data prefetching from start.  
I0408 23:52:44.983633 11898 data\_layer.cpp:195] Restarting data prefetching from start.  
I0408 23:53:04.592311 11923 data\_layer.cpp:195] Restarting data prefetching from start.  
I0408 23:53:24.199542 11941 data\_layer.cpp:195] Restarting data prefetching from start.  
I0408 23:53:43.808826 11974 data\_layer.cpp:195] Restarting data prefetching from start.  
I0408 23:54:03.417126 11989 data\_layer.cpp:195] Restarting data prefetching from start.  
I0408 23:54:23.025939 12008 data\_layer.cpp:195] Restarting data prefetching from start.  
I0408 23:54:35.630164 11493 solver.cpp:209] Iteration 300, loss = 0.00187102  
I0408 23:54:35.630205 11493 solver.cpp:224]     Train net output #0: loss = 0.00187103 (\* 1 = 0.00187103 loss)  
I0408 23:54:35.630214 11493 solver.cpp:445] Iteration 300, lr = 0.001  
I0408 23:56:55.706265 11493 solver.cpp:209] Iteration 400, loss = 0.000129756  
I0408 23:56:55.706351 11493 solver.cpp:224]     Train net output #0: loss = 0.000129764 (\* 1 = 0.000129764 loss)  
I0408 23:56:55.706360 11493 solver.cpp:445] Iteration 400, lr = 0.001  
I0408 23:59:14.362998 11493 solver.cpp:264] Iteration 500, Testing net (#0)  
I0408 23:59:14.363026 11493 net.cpp:652] Copying source layer cifar  
I0408 23:59:14.363034 11493 net.cpp:652] Copying source layer conv1  
I0408 23:59:14.363039 11493 net.cpp:652] Copying source layer pool1  
I0408 23:59:14.363044 11493 net.cpp:652] Copying source layer relu1  
I0408 23:59:14.363049 11493 net.cpp:652] Copying source layer conv2  
I0408 23:59:14.363054 11493 net.cpp:652] Copying source layer relu2  
I0408 23:59:14.363059 11493 net.cpp:652] Copying source layer pool2  
I0408 23:59:14.363064 11493 net.cpp:652] Copying source layer conv3  
I0408 23:59:14.363068 11493 net.cpp:652] Copying source layer relu3  
I0408 23:59:14.363073 11493 net.cpp:652] Copying source layer pool3  
I0408 23:59:14.363077 11493 net.cpp:652] Copying source layer ip1  
I0408 23:59:14.363082 11493 net.cpp:652] Copying source layer ip2  
I0408 23:59:14.363088 11493 net.cpp:652] Copying source layer loss  
I0408 23:59:14.364789 12279 data\_layer.cpp:195] Restarting data prefetching from start.  
I0409 00:00:10.128435 11493 solver.cpp:315]     Test net output #0: accuracy = 0.99  
I0409 00:00:10.128469 11493 solver.cpp:315]     Test net output #1: loss = 0.0209001 (\* 1 = 0.0209001 loss)  
I0409 00:00:11.528825 11493 solver.cpp:209] Iteration 500, loss = 0.000202642  
I0409 00:00:11.528863 11493 solver.cpp:224]     Train net output #0: loss = 0.000202651 (\* 1 = 0.000202651 loss)  
I0409 00:00:11.528873 11493 solver.cpp:445] Iteration 500, lr = 0.001

……

I0409 00:12:46.198276 11493 solver.cpp:315]     Test net output #0: accuracy = 0.99  
I0409 00:12:46.198317 11493 solver.cpp:315]     Test net output #1: loss = 0.0209753 (\* 1 = 0.0209753 loss)  
I0409 00:12:47.598284 11493 solver.cpp:209] Iteration 1000, loss = 0.000381682  
I0409 00:12:47.598323 11493 solver.cpp:224]     Train net output #0: loss = 0.00038169 (\* 1 = 0.00038169 loss)  
I0409 00:12:47.598333 11493 solver.cpp:445] Iteration 1000, lr = 0.001

训练完 将所要判断的图片输入，得出推算结果

prediction shape: (1, 2)  
['motobikes']  
[1]

# [学习笔记：深度学习网络特征逐层可视化](http://blog.csdn.net/lynnandwei/article/details/45716849)

分类： [Image processing](http://blog.csdn.net/lynnandwei/article/category/1185403) [Machine Learning](http://blog.csdn.net/lynnandwei/article/category/2864799)2015-05-14 15:23 167人阅读 [评论](http://blog.csdn.net/lynnandwei/article/details/45716849#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/lynnandwei/article/details/45716849#report)

“         Deep Learning很吸引人，也很玄乎的一个点就是大家都说它可以提取到分级的逐层抽象的特征。但对我们来说，总是耳听为虚，眼见为实。所以，每当我们训练完一个深度模型后，我们还特别想把这个深度模型学到的东西给可视化出来，好弄明白它到底学到了什么东西，是不是有意义的，是不是像传说中的那样神奇”    zouxy09   http://blog.csdn.net/zouxy09/article/details/10012747  
  
于是在训练完自己的数据之后，也来看看 每一层的参数 及 输出 是否有明显的可视化效果。

由于系统里 无法show的问题，我改成了将结果存成图片，然后从服务器上拉到本地来看。对应的函数做了一点小的修改。

def vis\_square(data, padsize=1, padval=0):  
    data -= data.min()  
    data /= data.max()  
  
  
    # force the number of filters to be square  
    n = int(np.ceil(np.sqrt(data.shape[0])))  
    padding = ((0, n \*\* 2 - data.shape[0]), (0, padsize), (0, padsize)) + ((0, 0),) \* (data.ndim - 3)  
    data = np.pad(data, padding, mode='constant', constant\_values=(padval, padval))  
  
  
    # tile the filters into an image  
    data = data.reshape((n, n) + data.shape[1:]).transpose((0, 2, 1, 3) + tuple(range(4, data.ndim + 1)))  
    data = data.reshape((n \* data.shape[1], n \* data.shape[3]) + data.shape[4:])  
    #pylab.imshow(data)  
    #pylab.show()  
    plt.figure()  
    plt.imshow(data)  
    plt.savefig("t.jpg")

原图 缩放到32\*32后如下

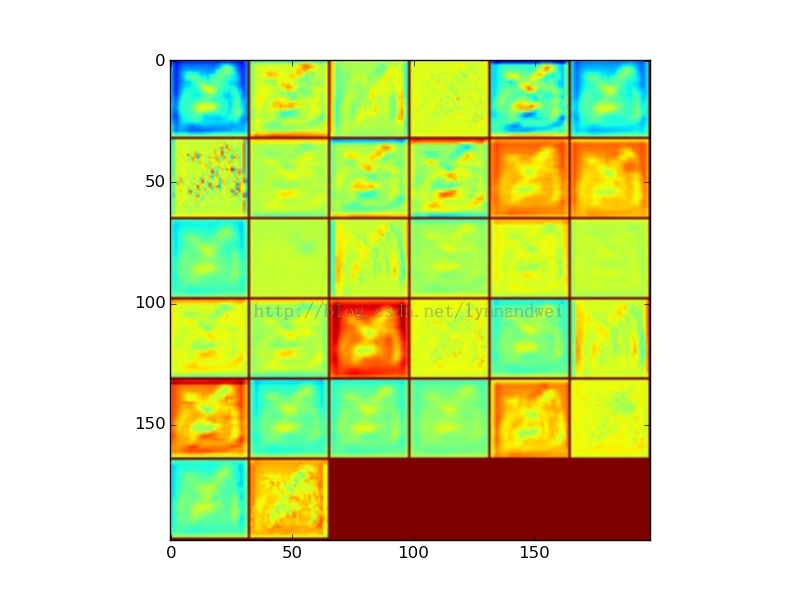


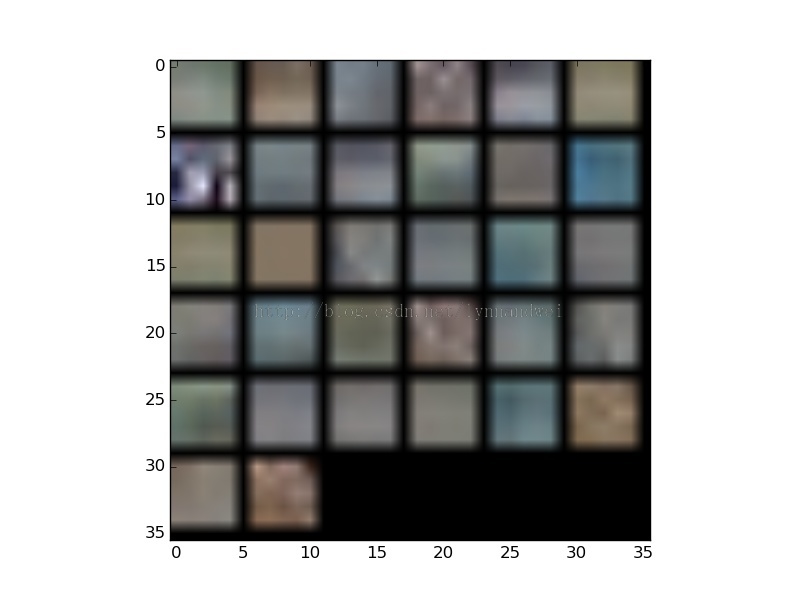
输出代码如下 ：

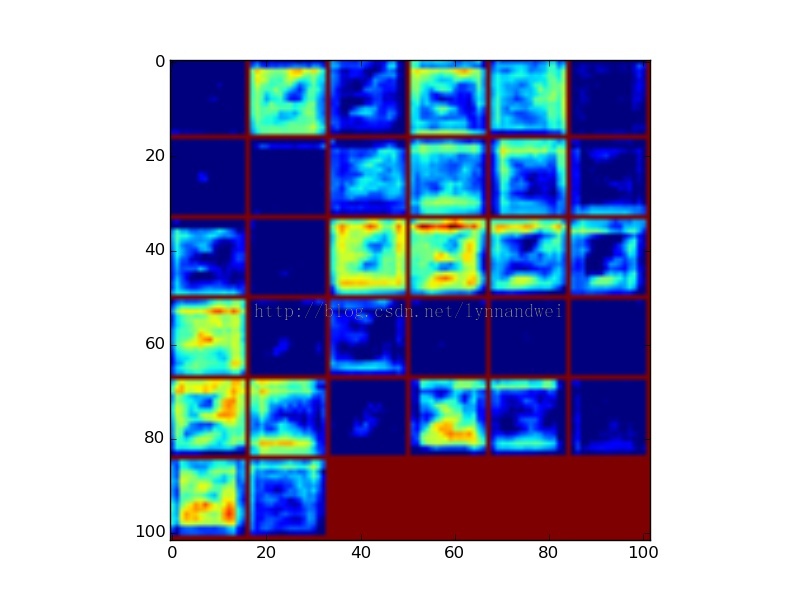
觉得输出结果还是有明显的可视性。

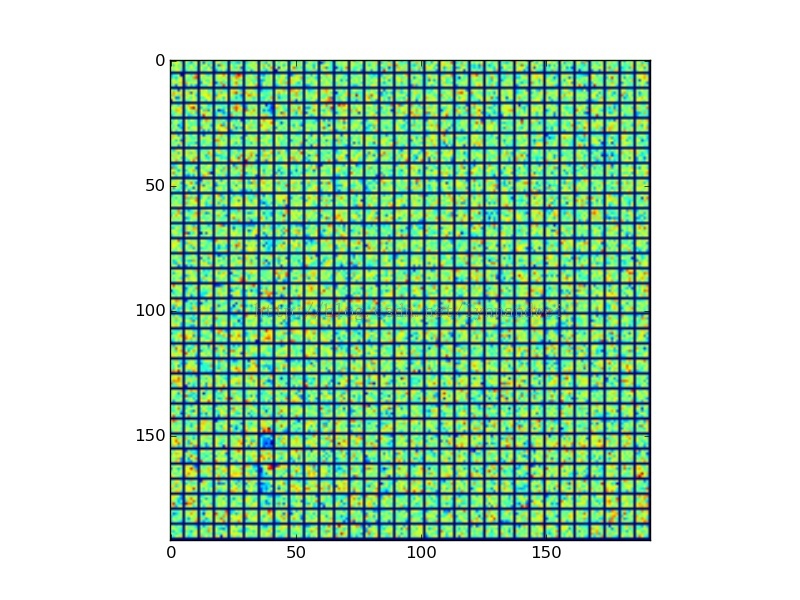
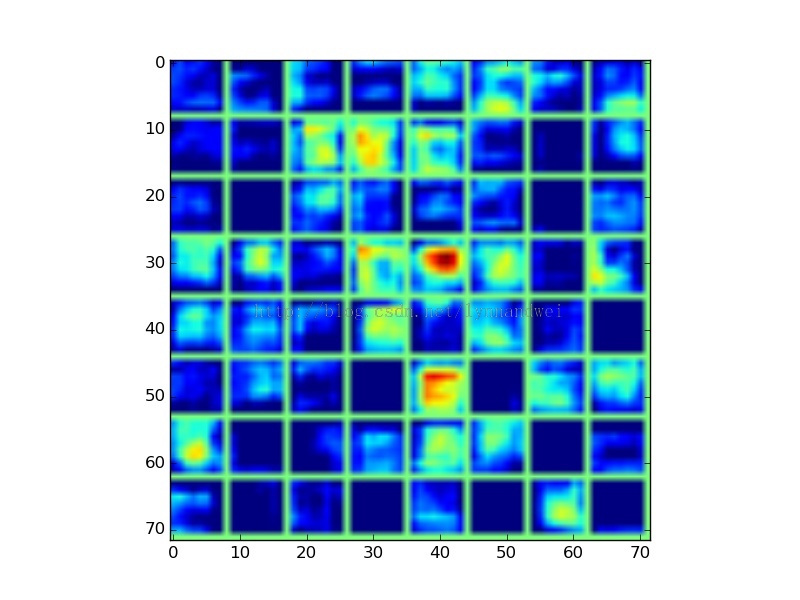
feat = net.blobs['conv1'].data[0]  
vis\_square(feat, padval=1)

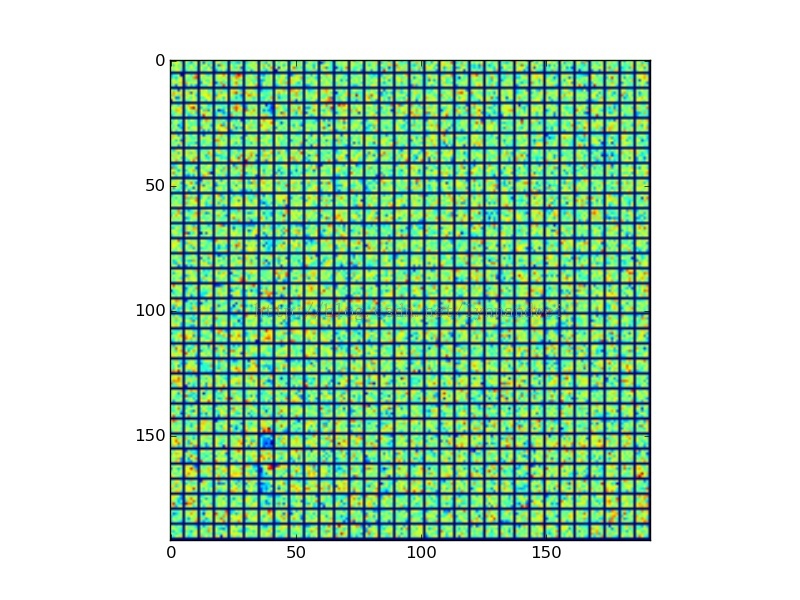
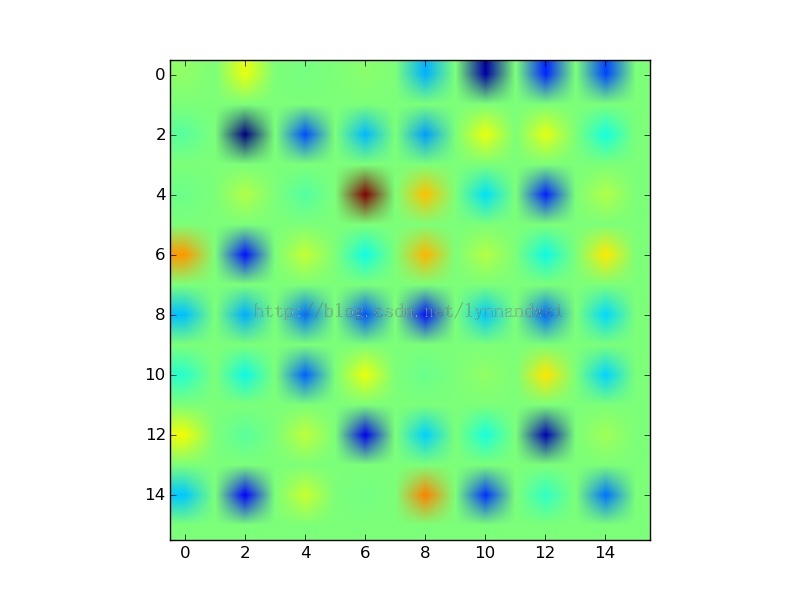
filters = net.params['conv1'][0].data  
vis\_square(filters.transpose(0,2,3,1))



  
  
  
feat = net.blobs['conv2'].data[0]  
vis\_square(feat, padval=1)

filters = net.params['conv2'][0].data  
vis\_square(filters[:32].reshape(32\*\*2, 5, 5))  
  


  
  
  
  
  
  
feat = net.blobs['conv3'].data[0]  
vis\_square(feat, padval=0.5)  
  
feat = net.params['conv3'][0].data  
vis\_square(filters[:32].reshape(32\*\*2,5,5))  
  


  
  
feat = net.blobs['ip1'].data[0]  
vis\_square(feat)  
  
  
feat = net.blobs['ip2'].data[0]  
vis\_square(feat)

