



转

## 卷积神经网络Lenet-5详解

2017年03月31日 18:14:09

阅读量：1.

### 卷积神经网络Lenet-5实现

原文地址：<http://blog.csdn.net/hjimce/article/details/47323463>

作者：hjimce

卷积神经网络**算法**是n年前就有的算法，只是近年来因为**深度学习**相关算法为多层网络的训练提供了新方法，然后现在电脑的计算能力已非当年的那种计算水平，同时现在的训练数据很多，于是神经网络的相关算法又重新火了起来，因此卷积神经网络就又活了起来。

在开始前，我们需要明确的是网上讲的卷积神经网络的相关教程一般指的是神经网络的前向传导过程，反向传播都是用梯度下降法进行训练，大部分深度学习库，都已经把反向求导的功能给封装好了，如果想要深入学习反向求导，就需要自己慢慢学了。

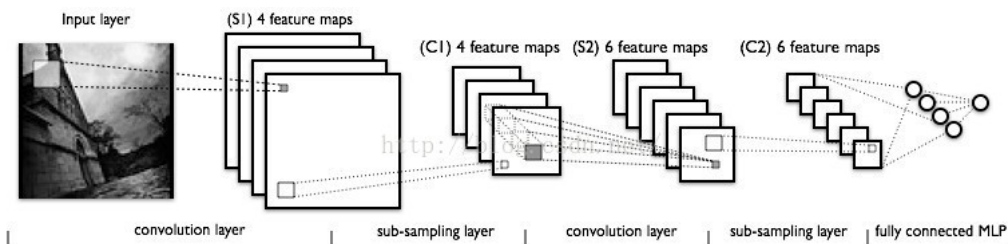
因为卷积神经网络的经典模型是：Lenet-5实现，只要理解了这个的前向传导过程，基本上就OK了，因此我们后面主要讲解Lenet-5的实现。

#### 一、理论阶段

作为CNN的入门文章，没有打算啰嗦太多的东西，因为什么权值共享、局部感受野什么的，讲那么多，都是那些生物学的相关理论，看了那些玩意，大部分初学者已经烦了。卷积神经网络的相关博文也是一大堆，但是讲的，基本上都是抄过来抄过去，就像我之前不理解从S2层到C3层是怎么实现的，网上看了一大堆教程，没有一个解答这个问题的。我的个人感觉整个过程，就只有S2到C3是最难理解的。接着我将用最浅显易懂的方式进行讲解。

##### 1、卷积

卷积的概念这个我想只要学过图像处理的人都懂的概念了，这个不解释。我们知道对于给定的一幅图像来说，给定一个卷积核，卷积就是根据卷积窗口，进行像素的加权求和。



卷积神经网络与我们之前所学到的图像的卷积的区别，我的理解是：我们之前学图像处理遇到卷积，一般来说，这个卷积核是已知的，比如各种边缘检测算子、高斯模糊等这些，都是已经知道卷积核，然后再与图像进行卷积运算。然而深度学习中的卷积神经网络卷积核是未知的，我们训练一个神经网络，就是要训练得出这些卷积核，而这些卷积核就相当于我们学单层感知器的时候的那些参数W，因此你可以把这些待学习的卷积核看成是神经网络的训练参数W。

##### 2、池化

刚开始学习CNN的时候，看到这个词，好像高大上的样子，于是查了很多资料，理论一大堆，但是实践、算法实现却看不到，也不懂池化要怎么实现？其实所谓的池化，就是图片下采样。这个时候，你会发现CNN每一层的构建跟图像高斯金字塔的构建有点类似，因此你如果已经懂得了图像金字塔融合的相关算法，那么就变的容易理解了。在高斯金字塔构建中，层通过卷积，然后卷积后进行下采样，而CNN也是同样的过程。废话不多说，这里就讲一下，CNN的池化：

CNN的池化(图像下采样)方法很多：Mean pooling(均值采样)、Max pooling(最大值采样)、Overlapping (重叠采样)、L2 pooling(均方采样)、Local Contrast Normalization(归一化采样)、Stochastic pooling(随即采样)、Def-pooling(形变约束采样)其中最经典的是最大池化，因此我就解释一下最大池化的实现：

10	15	80	12
18	30	11	20
11	19	48	51
30	31	45	59

原图片

为了简单起见，我用上面的图片作为例子，假设上面的图片大小是4\*4的，如上图所示，然后图片中每个像素点的值是各个格子中的数值。然后我要对这张4\*4的图片进行池化，池化的大小为(2,2)，跨步为2，那么采用最大池化也就是对上面4\*4的图片进行分块，每个块的大小为2\*2，然后统计每个块的最大值，作为下采样后图片的像素值，具体计算如下图所示：

$\max(10,15,18,30)=30$	$\max(80,12,11,20)=80$
$\max(11,19,30,31)=31$	$\max(48,51,45,59)=59$

也就是说我们最后得到下采样后的图片为：

30	80
31	59

这就是所谓的最大池化。当然以后你还会遇到各种池化方法，比如均值池化，也就是对每个块求取平均值作为下采样的新像素值。还有重叠采样的池化，我上面这个例子是没有重叠的采样的，也就是每个块之间没有相互重叠的部分，上面我说的跨步为2，就是为了使得分块都非重叠，等等，这些以后再跟大家解释池化常用方法。这里就先记住最大池化就好了，因为这个目前是最常用的。

### 3、feature maps

这个单词国人把它翻译成特征图，挺起来很专业的名词。那么什么叫特征图呢？特征图其实说白了就是CNN中的每张图片，都可以称之为特征图张。在CNN中，我们要训练的卷积核并不是仅仅只有一个，这些卷积核用于提取特征，卷积核个数越



联系我们



关于 招聘  
©2018 CSDN  
百度提供

经营性网站备案  
网络110报警  
中国互联网  
北京互联网

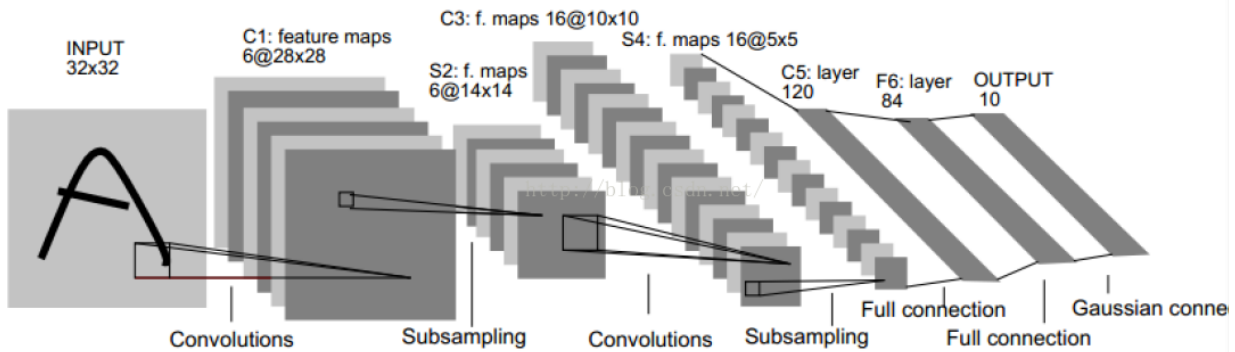
多，提取的特征越多，理论上来说精度也会更高，然而卷积核一堆，意味着我们要训练的参数的个数越多。在LeNet-5结构中，第一层卷积核选择了6个，而在AlexNet中，第一层卷积核就选择了96个，具体多少个合适，还有待学习。

回到特征图概念，CNN的每一个卷积层我们都要人为的选取合适的卷积核个数，及卷积核大小。每个卷积核与图片进行卷积，就可以得到一张特征图了，比如LeNet-5经典结构中，第一层卷积核选择了6个，我们可以得到6个特征图，这些特征图也就是下一层网络的输入了。我们也可以把输入图片看成一张特征图，作为第一层网络的输入。

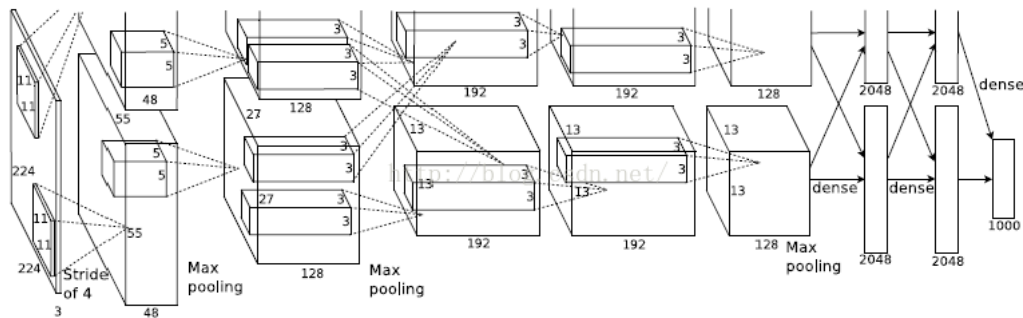
#### 4、CNN的经典结构

对于刚入门CNN的人来说，我们首先需要现在的一些经典结构：

(1)LeNet-5。这个是n多年前就有的一个CNN的经典结构，主要是用于手写字体的识别，也是刚入门需要学习熟悉的一个网络，我的这篇博文主要就是要讲这个网络



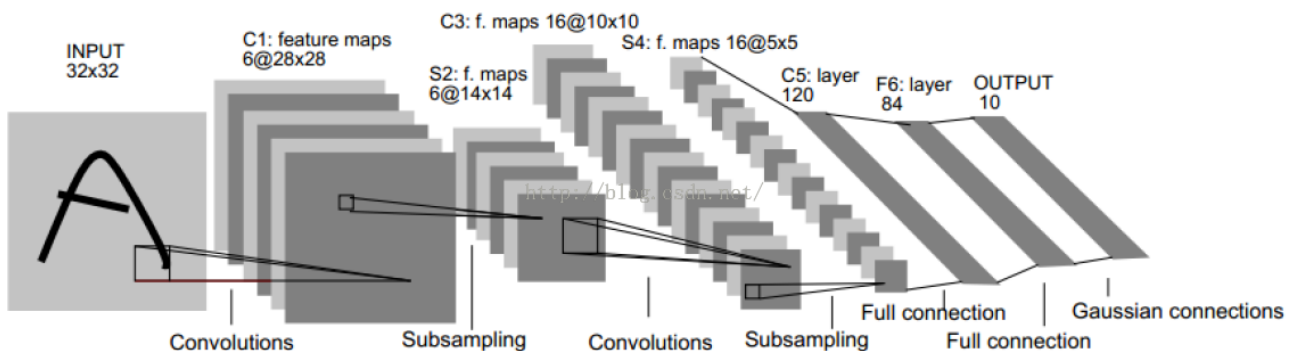
(2)AlexNet.



在imagenet上的图像分类challenge上大神Alex提出的alexnet网络结构模型赢得了2012届的冠军，振奋人心，利用CNN实现了图片分类，别人用传统的机器学习算法调参跳到半死也就那样，Alex利用CNN精度远超传统的网络。

其它的还有什么《Network In Network》，GoogLeNet、Deconvolution Network，在以后的学习中我们会遇到。比如利用Deconvolution Network反卷积网络实现图片的去模糊，牛逼哄哄。

OK，理论阶段就啰嗦到这里就好了，接着就讲解 LeNet-5，LeNet-5是用于手写字体的识别的一个经典CNN：



LeNet-5结构

输入：32\*32的手写字体图片，这些手写字体包含0~9数字，也就是相当于10个类别的图片

输出：分类结果，0~9之间的一个数



联系我们



关于 招聘  
©2018 CSDN  
百度提供

经营性网站备案  
网络110报警  
中国互联网  
北京互联网

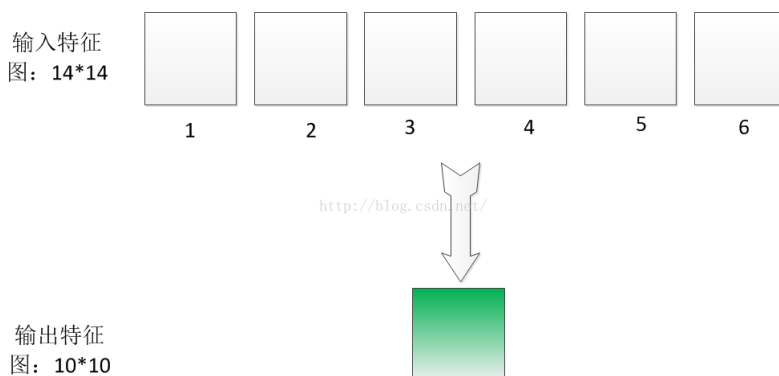
因此我们可以知道，这是一个多分类问题，总共有十个类，因此神经网络的最后输出层必然是SoftMax问题，然后神经元的个数是10个。LeNet-5结构：

输入层：32\*32的图片，也就是相当于1024个神经元

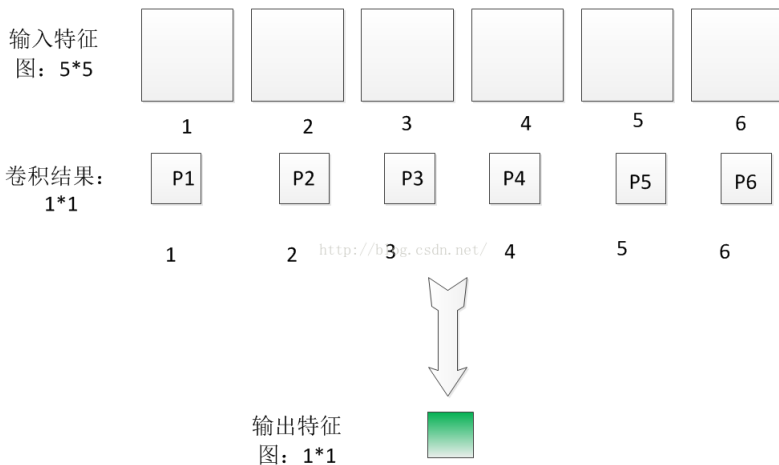
**C1层**：paper作者，选择6个特征卷积核，然后卷积核大小选择5\*5，这样我们可以得到6个特征图，然后每个特征图的大小为32-5+1=28，也就是神经元的个数为6\*28\*28=784。

**S2层**：这就是下采样层，也就是使用最大池化进行下采样，池化的size，选择(2,2)，也就是相当于对C1层28\*28的图片进行分块，每个块的大小为2\*2，这样我们可以得到14\*14个块，然后我们统计每个块中，最大的值作为下采样的新像素，我们可以得到S1结果为：14\*14大小的图片，共有6个这样的图片。

**C3层**：卷积层，这一层我们选择卷积核的大小依旧为5\*5，据此我们可以得到新的图片大小为14-5+1=10，然后我们得到16张特征图。那么问题来了？这一层是最难理解的，我们知道S2包含：6张14\*14大小的图片，我们希望这一层的结果是：16张10\*10的图片。这16张图片的每一张，是通过S2的6张图片进行加权组合得到的，具体是怎么组合的问题如下图所示：



为了解释这个问题，我们先从简单的开始，我现在假设输入6特征图的大小是5\*5的，分别用6个5\*5的卷积核进行卷积，得到6个卷积结果图片大小为1\*1，如下图所示：



为了简便起见，我这里先做一些标号的定义：我们假设输入第i个特征图的各个像素值为 $x_{1i}, x_{2i}, \dots, x_{25i}$ ，因为每个特征图有25个像素。因此第i个特征图经过5\*5的图片卷积后，得到的卷积结果图片的像素值 $P_i$ 可以表示成：

$$P_i = w_{1i} \cdot x_{1i} + w_{2i} \cdot x_{2i} + \dots + w_{25i} \cdot x_{25i}$$

这个是卷积公式，不解释。因此对于上面的P1~P6的计算方法，这个就是直接根据公式。然后我们把P1~P6相加起来，也就是：

$$P = P_1 + P_2 + \dots + P_6$$

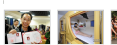
把上面的P的计算公式，代入上式，那么我们可以得到：

$$P = WX$$

其中X就是输入的那6张5\*5特征图片的各个像素点值，而W就是我们需要学习的参数，也就相当于6个5\*5的卷积核，当然它包含着6\*(5\*5)个参数。因此我们的输出特征图就是：



家用电梯



联系我们



关于 招聘  
©2018 CSDN  
百度提供

经营性网站备案  
网络110报警  
中国互联网  
北京互联网



收藏



评论



微信



微博



QQ

$$\text{Out} = f(P + b)$$

这个就是从S2到C3的计算方法，其中b表示偏置项，f为激活函数。

我们回归到原来的问题：有6张输入14\*14的特征图片，我们希望用5\*5的卷积核，然后最后我们希望得到一张10\*10的卷积特征图？

根据上面的过程，也就是其实我们用5\*5的卷积核去卷积每一张输入的特征图，当然每张特征图的卷积核参数是不一样的，也就是不共享，因此我们就相当于需要6\*(5\*5)个参数。对每一张输入特征图进行卷积后，我们得到6张10\*10，新图片，时候，我们把这6张图片相加在一起，然后加一个偏置项b，然后用激活函数进行映射，就可以得到一张10\*10的输出特征图了。

而我们希望得到16张10\*10的输出特征图，因此我们就需要卷积参数个数为16\*(6\*(5\*5))=16\*6\*(5\*5)个参数。总之，每个图片是通过S2图片进行卷积后，然后相加，并且加上偏置b，最后在进行激活函数映射得到的结果。

**S4层**：下采样层，比较简单，也是知己对C3的16张10\*10的图片进行最大池化，池化块的大小为2\*2。因此最后S4层大小为5\*5的图片。至此我们的神经元个数已经减少为：16\*5\*5=400。

**C5层**：我们继续用5\*5的卷积核进行卷积，然后我们希望得到120个特征图。这样C5层图片的大小为5-5+1=1，也就是1个神经元，120个特征图，因此最后只剩下120个神经元了。这个时候，神经元的个数已经够少的了，后面我们就可以直接利用全连接神经网络，进行这120个神经元的后续处理，后面具体要怎么搞，只要懂多层感知器的都懂了，不解释。

上面的结构，只是一种参考，在现实使用中，每一层特征图需要多少个，卷积核大小选择，还有池化的时候采样率要多少这些都是变化的，这就是所谓的CNN调参，我们需要学会灵活多变。

比如我们可以把上面的结构改为：C1层卷积核大小为7\*7，然后把C3层卷积核大小改为3\*3等，然后特征图的个数也是自选，说不定得到手写字体识别的精度比上面那个还高，这也是有可能的，总之一句话：需要学会灵活多变，需要学会CNN调参。

## 二、实战阶段

学习CNN的源码实现网站：<http://deeplearning.net/tutorial/lenet.html#lenet>

### 1、训练数据获取

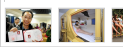
在theano学习库中有手写字体的库，可以从网上下载到，名为：mnist.pkl.gz的手写字体库，里面包含了三个部分的数据，训练数据集train\_set：50000个训练样本，验证集valid\_set，我们可以用如下的代码读取这些数据，然后用plot显示其中的一张图片：

```
[python]
01. <span style="font-size:18px;">import cPickle
02. import gzip
03. import numpy as np
04. import matplotlib.pyplot as plt
05. f = gzip.open('mnist.pkl.gz', 'rb')
06. train_set, valid_set, test_set = cPickle.load(f)
07. f.close()
08. tx,ty=train_set;
09.
10. #查看训练样本
11. print np.shape(tx)#可以看到tx大小为（50000,28*28）的二维矩阵
12. print np.shape(ty)#可以看到ty大小为（50000,1）的矩阵
13. #图片显示
14. A=tx[8].reshape(28,28)#第八个训练样本
15. Y=ty[8]
16. print Y
17. plt.imshow(A,cmap='gray')#显示手写字体图片</span>
```

在上面的代码中我显示的是第8张图片，可以看到如下结果：



家用电梯



联系我们



关于 招聘

©2018 CSDN

百度提供

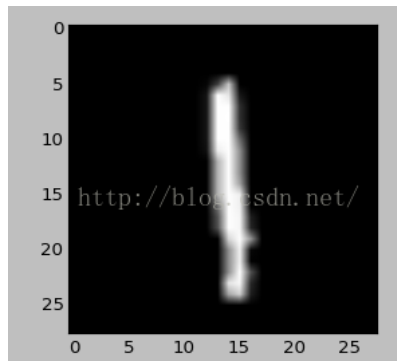
经营性网站备案

网络110报警

中国互联网网

北京互联网通





联系我们



关于 招聘  
©2018 CSDN  
百度提供

经营性网站备案  
网络110报警  
中国互联网  
北京互联网

第八个样本是数字1。

## 2、LeNet-5实现

首先你要知道mnist.pkl.gz这个库给我们的图片的大小是28\*28的，因此我们可以第一步选择5\*5的卷积核进行卷积得到4，同时我们希望C1层得到20张特征图，等等，具体的代码实现如下；

```
[python]
01. import os
02. import sys
03. import timeit
04.
05. import numpy
06.
07. import theano
08. import theano.tensor as T
09. from theano.tensor.signal import downsample
10. from theano.tensor.nnet import conv
11.
12. from logistic_sgd import LogisticRegression, load_data
13. from mlp import HiddenLayer
14.
15. #卷积神经网络的一层，包含：卷积+下采样两个步骤
16. #算法的过程是：卷积->下采样->激活函数
17. class LeNetConvPoolLayer(object):
18.
19.     #image_shape是输入数据的相关参数设置 filter_shape本层的相关参数设置
20.     def __init__(self, rng, input, filter_shape, image_shape, poolsize=(2, 2)):
21.         """
22.         :type rng: numpy.random.RandomState
23.         :param rng: a random number generator used to initialize weights
24.
25.         3、input: 输入特征图数据，也就是n幅特征图片
26.
27.         4、参数 filter_shape: (number of filters, num input feature maps,
28.                                filter height, filter width)
29.         num of filters: 是卷积核的个数，有多少个卷积核，那么本层的out feature maps的个数
30.         也将生成多少个。num input feature maps: 输入特征图的个数。
31.         然后接着filter height, filter width是卷积核的宽高，比如5*5,9*9.....
32.         filter_shape是列表，因此我们可以用filter_shape[0]获取卷积核个数
33.
34.         5、参数 image_shape: (batch size, num input feature maps,
35.                                image height, image width),
36.         batch size: 批量训练样本个数，num input feature maps: 输入特征图的个数
37.         image height, image width分别是输入的feature map图片的大小。
38.         image_shape是一个列表类型，所以可以直接用索引，访问上面的4个参数，索引下标从
39.         0~3。比如image_shape[2]=image_height image_shape[3]=num input feature maps
40.
41.         6、参数 poolsize: 池化下采样的块大小，一般为(2,2)
42.         """
43.
44.         assert image_shape[1] == filter_shape[1] #判断输入特征图的个数是否一致，如果不一致是错误的
45.         self.input = input
46.
47.         # fan_in=num input feature maps *filter height*filter width
48.         #numpy.prod(x)函数为计算x各个元素的乘积
49.         #也就是说fan_in就相当于每个即将输出的feature map所需要链接参数数值的个数
50.         fan_in = numpy.prod(filter_shape[1:])
51.         # fan_out=num output feature maps * filter height * filter width
52.         fan_out = (filter_shape[0] * numpy.prod(filter_shape[2:])) /
53.             numpy.prod(poolsize))
54.         # 把参数初始化到[-a,a]之间的数，其中a=sqrt(6./(fan_in + fan_out)),然后参数采用均匀采样
55.         #权值需要多少个？卷积核个数*输入特征图个数*卷积核宽*卷积核高？这样没有包含采样层的链接权值个数
56.         W_bound = numpy.sqrt(6. / (fan_in + fan_out))
57.         self.W = theano.shared(
58.             numpy.asarray(
59.                 rng.uniform(low=-W_bound, high=W_bound, size=filter_shape),
```

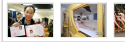
```

60.         dtype=theano.config.floatX
61.     ),
62.     borrow=True
63. )
64.
65.     # b为偏置，是一维的向量。每个输出特征图i对应一个偏置参数b[i]
66.     #,因此下面初始化b的个数就是特征图的个数filter_shape[0]
67.     b_values = numpy.zeros((filter_shape[0],), dtype=theano.config.floatX)
68.     self.b = theano.shared(value=b_values, borrow=True)
69.
70.     # 卷积层操作，函数conv.conv2d的第一个参数为输入的特征图，第二个参数为随机出事的卷积核参数
71.     #第三个参数为卷积核的相关属性，输入特征图的相关属性
72.     conv_out = conv.conv2d(
73.         input=input,
74.         filters=self.W,
75.         filter_shape=filter_shape,
76.         image_shape=image_shape
77.     )
78.
79.     # 池化操作，最大池化
80.     pooled_out = downsample.max_pool_2d(
81.         input=conv_out,
82.         ds=poolsize,
83.         ignore_border=True
84.     )
85.     #激励函数，也就是说是先经过卷积核再池化后，然后在进行非线性映射
86.     # add the bias term. Since the bias is a vector (1D array), we first
87.     # reshape it to a tensor of shape (1, n_filters, 1, 1). Each bias will
88.     # thus be broadcasted across mini-batches and feature map
89.     # width & height
90.     self.output = T.tanh(pooled_out + self.b.dimshufle('x', 0, 'x', 'x'))
91.
92.     # 保存参数
93.     self.params = [self.W, self.b]
94.     self.input = input
95.
96. #测试函数
97. def evaluate_lenet5(learning_rate=0.1, n_epochs=200,
98.                    dataset='mnist.pkl.gz',
99.                    nkerns=[20, 50], batch_size=500):
100.     """ Demonstrates lenet on MNIST dataset
101.
102.
103.     :learning_rate: 梯度下降法的学习率
104.
105.     :n_epochs: 最大迭代次数
106.
107.     :type dataset: string
108.     :param dataset: path to the dataset used for training /testing (MNIST here)
109.
110.     :nkerns: 每个卷积层的卷积核个数，第一层卷积核个数为 nkerns[0]=20,第二层卷积核个数
111.     为50个
112.     """
113.
114.     rng = numpy.random.RandomState(23455)
115.
116.     datasets = load_data(dataset)#加载训练数据，训练数据包含三个部分
117.
118.     train_set_x, train_set_y = datasets[0]#训练数据
119.     valid_set_x, valid_set_y = datasets[1]#验证数据
120.     test_set_x, test_set_y = datasets[2]#测试数据
121.
122.     # 计算批量训练可以分多少批数据进行训练，这个只要是知道批量训练的人都知道
123.     n_train_batches = train_set_x.get_value(borrow=True).shape[0]#训练数据个数
124.     n_valid_batches = valid_set_x.get_value(borrow=True).shape[0]
125.     n_test_batches = test_set_x.get_value(borrow=True).shape[0]
126.     n_train_batches /= batch_size#批数
127.     n_valid_batches /= batch_size
128.     n_test_batches /= batch_size
129.
130.     # allocate symbolic variables for the data
131.     index = T.lscalar() # index to a [mini]batch
132.
133.     # start-snippet-1
134.     x = T.matrix('x') # the data is presented as rasterized images
135.     y = T.ivector('y') # the labels are presented as 1D vector of
136.     # [int] labels
137.
138.
139.     # Reshape matrix of rasterized images of shape (batch_size, 28 * 28)
140.     # to a 4D tensor, compatible with our LeNetConvPoolLayer
141.     # (28, 28) is the size of MNIST images.
142.     layer0_input = x.reshape((batch_size, 1, 28, 28))
143.
144.     '''构建第一层网络:
145.     image_shape: 输入大小为28*28的特征图，batch_size个训练数据，每个训练数据有1个特征图
146.     filter_shape: 卷积核个数为nkerns[0]=20，因此本层每个训练样本即将生成20个特征图

```



家用电梯



联系我们



关于 招聘  
©2018 CSDN  
百度提供

经营性网站备案  
网络110报警  
中国互联网  
北京互联网

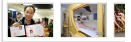
```

147. 经过卷积操作，图片大小变为 $(28-5+1, 28-5+1) = (24, 24)$ 
148. 经过池化操作，图片大小变为 $(24/2, 24/2) = (12, 12)$ 
149. 最后生成的本层image_shape为(batch_size, nkerns[0], 12, 12)'''
150. layer0 = LeNetConvPoolLayer(
151.     rng,
152.     input=layer0_input,
153.     image_shape=(batch_size, 1, 28, 28),
154.     filter_shape=(nkerns[0], 1, 5, 5),
155.     poolsize=(2, 2)
156. )
157.
158. '''构建第二层网络：输入batch_size个训练图片，经过第一层的卷积后，每个训练图片有nkerns[0]个特征
图，每个特征图
159. 大小为12*12
160. 经过卷积后，图片大小变为 $(12-5+1, 12-5+1) = (8, 8)$ 
161. 经过池化后，图片大小变为 $(8/2, 8/2) = (4, 4)$ 
162. 最后生成的本层的image_shape为(batch_size, nkerns[1], 4, 4)'''
163. layer1 = LeNetConvPoolLayer(
164.     rng,
165.     input=layer0.output,
166.     image_shape=(batch_size, nkerns[0], 12, 12),
167.     filter_shape=(nkerns[1], nkerns[0], 5, 5),
168.     poolsize=(2, 2)
169. )
170.
171. # the HiddenLayer being fully-connected, it operates on 2D matrices of
172. # shape (batch_size, num_pixels) (i.e matrix of rasterized images).
173. # This will generate a matrix of shape (batch_size, nkerns[1] * 4 * 4),
174. # or (500, 50 * 4 * 4) = (500, 800) with the default values.
175. layer2_input = layer1.output.flatten(2)
176.
177. '''全链接：输入layer2_input是一个二维的矩阵，第一维表示样本，第二维表示上面经过卷积下采样后
每个样本所得到的神经元，也就是每个样本的特征，HiddenLayer类是一个单层网络结构
178. 下面的layer2把神经元个数由800个压缩映射为500个'''
179. layer2 = HiddenLayer(
180.     rng,
181.     input=layer2_input,
182.     n_in=nkerns[1] * 4 * 4,
183.     n_out=500,
184.     activation=T.tanh
185. )
186.
187.
188. # 最后一层：逻辑回归层分类判别，把500个神经元，压缩映射成10个神经元，分别对应于手写字体的0~9
189. layer3 = LogisticRegression(input=layer2.output, n_in=500, n_out=10)
190.
191. # the cost we minimize during training is the NLL of the model
192. cost = layer3.negative_log_likelihood(y)
193.
194. # create a function to compute the mistakes that are made by the model
195. test_model = theano.function(
196.     [index],
197.     layer3.errors(y),
198.     givens={
199.         x: test_set_x[index * batch_size: (index + 1) * batch_size],
200.         y: test_set_y[index * batch_size: (index + 1) * batch_size]
201.     }
202. )
203.
204. validate_model = theano.function(
205.     [index],
206.     layer3.errors(y),
207.     givens={
208.         x: valid_set_x[index * batch_size: (index + 1) * batch_size],
209.         y: valid_set_y[index * batch_size: (index + 1) * batch_size]
210.     }
211. )
212.
213. #把所有的参数放在同一个列表里，可直接使用列表相加
214. params = layer3.params + layer2.params + layer1.params + layer0.params
215.
216. #梯度求导
217. grads = T.grad(cost, params)
218.
219. # train_model is a function that updates the model parameters by
220. # SGD Since this model has many parameters, it would be tedious to
221. # manually create an update rule for each model parameter. We thus
222. # create the updates list by automatically looping over all
223. # (params[i], grads[i]) pairs.
224. updates = [
225.     (param_i, param_i - learning_rate * grad_i)
226.     for param_i, grad_i in zip(params, grads)
227. ]
228.
229. train_model = theano.function(
230.     [index],
231.     cost,
232.     updates=updates,

```



家用电梯



## 联系我们



关于 招聘  
©2018 CSDN  
百度提供

经营性网站  
网络110报警  
中国互联网  
北京互联网



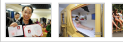
```

233.         givens={
234.             x: train_set_x[index * batch_size: (index + 1) * batch_size],
235.             y: train_set_y[index * batch_size: (index + 1) * batch_size]
236.         }
237.     )
238. # end-snippet-1
239.
240. #####
241. # TRAIN MODEL #
242. #####
243. print '... training'
244. # early-stopping parameters
245. patience = 10000 # look as this many examples regardless
246. patience_increase = 2 # wait this much longer when a new best is
247.                       # found
248. improvement_threshold = 0.995 # a relative improvement of this much is
249.                               # considered significant
250. validation_frequency = min(n_train_batches, patience / 2)
251.                           # go through this many
252.                           # minibatches before checking the network
253.                           # on the validation set; in this case we
254.                           # check every epoch
255.
256. best_validation_loss = numpy.inf
257. best_iter = 0
258. test_score = 0.
259. start_time = timeit.default_timer()
260.
261. epoch = 0
262. done_looping = False
263.
264. while (epoch < n_epochs) and (not done_looping):
265.     epoch = epoch + 1
266.     for minibatch_index in xrange(n_train_batches):#每一批训练数据
267.
268.         cost_ij = train_model(minibatch_index)
269.         iter = (epoch - 1) * n_train_batches + minibatch_index
270.         if (iter + 1) % validation_frequency == 0:
271.
272.             # compute zero-one loss on validation set
273.             validation_losses = [validate_model(i) for i
274.                                 in xrange(n_valid_batches)]
275.             this_validation_loss = numpy.mean(validation_losses)
276.             print('epoch %i, minibatch %i/%i, validation error %f %%' %
277.                   (epoch, minibatch_index + 1, n_train_batches,
278.                    this_validation_loss * 100.))
279.
280.             # if we got the best validation score until now
281.             if this_validation_loss < best_validation_loss:
282.
283.                 #improve patience if loss improvement is good enough
284.                 if this_validation_loss < best_validation_loss * \
285.                     improvement_threshold:
286.                     patience = max(patience, iter * patience_increase)
287.
288.                 # save best validation score and iteration number
289.                 best_validation_loss = this_validation_loss
290.                 best_iter = iter
291.
292.                 # test it on the test set
293.                 test_losses = [
294.                     test_model(i)
295.                     for i in xrange(n_test_batches)
296.                 ]
297.                 test_score = numpy.mean(test_losses)
298.                 print(('    epoch %i, minibatch %i/%i, test error of '
299.                       'best model %f %%') %
300.                       (epoch, minibatch_index + 1, n_train_batches,
301.                        test_score * 100.))
302.
303.             if patience <= iter:
304.                 done_looping = True
305.                 break
306.
307. end_time = timeit.default_timer()
308. print('Optimization complete.')
309. print('Best validation score of %f %% obtained at iteration %i, '
310.       'with test performance %f %%' %
311.       (best_validation_loss * 100., best_iter + 1, test_score * 100.))
312. print '>> sys.stderr, ('The code for file ' +
313.       os.path.split(__file__)[1] +
314.       ' ran for %.2fm' % ((end_time - start_time) / 60.))
315.
316. if __name__ == '__main__':
317.     evaluate_lenet5()
318.
319.

```



家用电梯



联系我们



关于 招聘  
©2018 CSDI  
百度提供

经营性网站  
网络110报警  
中国互联网  
北京互联网

```
320. | def experiment(state, channel):
321. |     evaluate_lenet5(state.learning_rate, dataset=state.dataset)
```

训练结果：

```
epoch 14, minibatch 100/100, test error of best model 1.550000 %
epoch 15, minibatch 100/100, validation error 1.670000 %
epoch 15, minibatch 100/100, test error of best model 1.500000 %
epoch 16, minibatch 100/100, validation error 1.620000 %
epoch 16, minibatch 100/100, test error of best model 1.440000 %
epoch 17, minibatch 100/100, validation error 1.590000 %
epoch 17, minibatch 100/100, test error of best model 1.410000 %
epoch 18, minibatch 100/100, validation error 1.570000 %
epoch 18, minibatch 100/100, test error of best model 1.420000 %
```

参考文献：

- 1、<http://blog.csdn.NET/zouxy09/article/details/8775360/>
- 2、<http://www.deeplearning.Net/tutorial/lenet.html#lenet>

\*\*\*\*\*作者：hjimce 时间：2015.8.6 联系QQ：1393852684 地址：<http://blog.csdn.net/hjimce> 转载请  
本行信息\*\*\*\*\*

个人分类：[CAFFE](#) [深度学习](#)



联系我们



关于 招聘  
©2018 CSDN  
百度提供

经营性网站  
网络110报警  
中国互联网  
北京互联网

想对作者说点什么？

## Tensorflow实例：（卷积神经网络）LeNet-5模型

Tensorflow实现LeNet-5模型，并用于mnist识别

XJY104165 2017-10-12 18:12:29 阅读数：5367

## 卷积神经网络（一）：LeNet5的基本结构

卷积神经网络LeNet5的结构介绍。

xuanyuansen 2014-12-08 10:44:43 阅读数：65703

## 经典卷积神经网络 之 LeNet-5 - CSDN博客

LeNet-5——1998 LeNet-5是卷积网络的开鼻祖,它是用来识别手写邮政编码的,论文可以参考Haffner. Gradient-based learning applied to document recognition. 大名...

2018-6-1

## Tensorflow实例:(卷积神经网络)LeNet-5模型 - CSDN博客

Tensorflow实现LeNet-5模型,并用于mnist识别... 本文通过LeNet-5模型,将给出卷积神经网络结构设计的一个通用模式。 LeNet-5模型 LeNet-5模型是Yann LeCun教授于1...

2018-6-5

## 耳鸣千万不可小视，几招教你解决...

上海华肤医院 · 顶新

## 深度学习 CNN卷积神经网络 LeNet-5详解



**卷积神经网络** ( Convolutional Neural Network, CNN ) : 是一种常见的深度学习架构, 受生物自然视觉认知机制(动物视觉皮层细胞负责检测光学信号)启发, 是一...

happyorg 2017-10-18 16:04:35 阅读数 : 5127

## 深度学习 CNN**卷积神经网络 LeNet-5**详解 - CSDN博客

**LeNet-5**是Yann LeCun在1998年设计的用于手写数字识别的**卷积神经网络**,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,它是早期**卷积神经网络**中最有代表性的...

2018-6-8

## **LeNet-5网络模型**详解 - CSDN博客

**卷积神经网络**在使用时往往是多层的,下面通过**LeNet-5**的网络连接来举例说明一个**卷积神经网络**的结构和特点。**LeNet-5**是Yann LeCun在1998年设计的用于手写数字识别的卷积...

2018-6-7

## 1.CNN学习笔记——理解结构, LeNet5介绍

**卷积神经网络**CNN的结构理解, LeNet5介绍

Kaido0 2016-11-14 17:49:29 阅读数 : 19205

## **LeNet5**的深入解析

论文 : Gradient-based-learning-applied-to-document-recognition参考 : <http://blog.csdn.net/strint/article/details/44163869>

zhangjunhit 2016-12-09 11:35:53 阅读数 : 10904

## **卷积神经网络CNN**的详细解读,及经典分类**网络LeNet5**的介绍 - CSDN博客

CNN的基础模块为**卷积流**,其包括四个部分:**卷积**,池化,非线性,批量归一化。**卷积**...解读:一般CNN**神经网络**池化层通常都是利用最大池化,但是在**LeNet5**中池化稍微复杂...

2018-6-3

## **lenet-5**的理解 - CSDN博客

**lenet-5**是**卷积神经网络**的一种典型的网络结构。以下试图解释以下该网络结构。卷积公式如下: 首先,输入是一张图片,这张图片是32\*32的。。然后用6个5\*5的卷积...

2018-5-4

## **lenet-5**的理解

**lenet-5**是**卷积神经网络**的一种典型的网络结构。以下试图解释以下该网络结构。卷积公式如下: 首先,输入是一张图片,这张图片是32\*32的。。然后用6个5\*5的卷积核,于是就有了C1,变成了...

q5390498 2016-12-20 20:39:16 阅读数 : 775

## 女性得了静脉曲张变成蚯蚓腿怎么办？用这方法坚持3个月全恢复！

水英电器 · 顶新

## **LeNet5**的深入解析 - CSDN博客

论文:Gradient-based-learning-applied-to-document-recognition参考:<http://blog.csdn.net/strint/article/details/44163869>**LeNet5** 这个**网络**虽然很小,但是它包含了...

2018-6-7

## **卷积神经网络的网络结构——LeNet5** - CSDN博客

**卷积神经网络**的网络结构——以**LeNet-5**为例 **卷积神经网络**是一种特殊的多层神经网络,像其它的神经网络一样,**卷积神经网络**也使用一种反向传播算法来进行训练,不同之...

2018-5-29

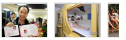
## 【DL笔记】**LeNet5**神经网络简介及TensorFlow实现

LeNet5神经网络简介

roguesir 2017-06-27 09:54:49 阅读数 : 10765



家用电梯



联系我们



关于 招聘  
©2018 CSDI  
百度提供

经营性网站  
网络110报警  
中国互联网  
北京互联网

## LeNet-5 理解

<http://blog.csdn.net/qiaofangjie/article/details/16826849> 点击打开链接

 huixingshao 2016-12-24 14:36:51 阅读数：586

## 卷积神经网络Lenet-5实现 - CSDN博客


原文地址:<http://blog.csdn.net/hjimce/article/details/47323463> 作者:hjimce 卷积神经网络算法是n年前就有的算法,只是近年来因为深度学习相关算法为多层的训练...

2018-6-12

undefined

## 卷积神经网络的网络结构——以LeNet-5为例

卷积神经网络是一种特殊的多层神经网络,像其它的神经网络一样,卷积神经网络也使用一种反向传播算法来进行训练,不同之处在于网络的结构。卷积神经网络的网络连接具有局部连接、参数共享的特点。局部连接是相对于普...

 strint 2015-03-09 23:24:32 阅读数：31442

## Deep Learning (深度学习) 学习笔记整理系列之LeNet-5卷积参数个人理解

Deep Learning (深度学习) 学习笔记整理系列的地址是<http://blog.csdn.net/zouxy09/article/details/8781543>,里面举了一个卷积例子用来说明参数...

 qiaofangjie 2013-11-19 17:06:54 阅读数：94152


## keras实现LeNet-5模型

先放一张LeNet-5模型的结构图关于LeNet-5模型每层具体的参数个数和如何连接本文不讲述,想了解的读者点击这个链接。下面的代码是使用keras实现,使用tensorflow为后端。#encoding:ut...

 csdn15698845876 2017-06-29 11:24:47 阅读数：2757

## LeNet-5模型详解及其TensorFlow代码实现

本文先分析了LeNet-5模型的结构,然后基于LeNet-5模型写了TensorFlow代码实现mnist数字识别,代码部分进行了详细注解,目前也在学习阶段,有错误欢迎指出,大家一起学习。 ...

 Enchanted\_ZhouH 2017-08-07 18:08:47 阅读数：6127

## caffe学习之Lenet-5详解

```
from pylab import * %matplotlib inline
caffe_root = './' # this file should be run from {caffe_ro...
```

 u011345885 2017-05-31 21:49:53 阅读数：786

## 50万码农评论：英语对于程序员有多重要？

不背单词和语法,一个公式学好英语



## LeNet-5网络结构解析

参考文章: 文章1 文章2 文章3 特殊性 神经元间的连接是非全连接的 同一层中某些神经元之间的连接的权重是共享的(即相同的) 权值共享 使用同一个Kernel池化 转:<http://blog...>

 acm\_fighting 2017-03-14 15:38:31 阅读数：4216

## 卷积神经网络Lenet-5实现

原文地址: <http://blog.csdn.net/hjimce/article/details/47323463> 作者: hjimce 卷积神经网络算法是n年前就有的算法,只是近年来因为深度学习相关算...

 XiaoXIANGZI222 2017-01-04 16:13:25 阅读数：1025

## 深度学习AlexNet模型详细分析

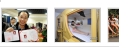
Alex在2012年提出的alexnet网络结构模型引爆了神经网络的应用热潮,并赢得了2012届图像识别大赛的冠军,使得CNN成为在图像分类上的核心算法模型。接下来本文对该网络配置结构中各个层进行...

 zyqdragon 2017-05-16 21:27:27 阅读数：26572

## Deep Learning学习之 卷积神经网络 (文字识别系统LeNet-5)



家用电梯



联系我们



关于 招聘

©2018 CSDN

百度提供

经营性网站

网络110报警

中国互联网

北京互联网

在经典的模式识别中，一般是事先提取特征。提取诸多特征后，要对这些特征进行相关性分析，找到最能代表字符的特征，去掉对分类无关和自相关的特征，这些特征的提取太过依赖人的经验和主观意识，提取到的特征的...

u013007900 2016-05-15 21:33:44 阅读数：12493

## lenet-5网络解析

lenet-5网络解析

kekong0713 2016-06-27 19:38:26 阅读数：1227

## 免费云主机试用一年

可以免费试用30天的移动云vps

百度广告



## LeNet5的基本结构

转自<http://blog.csdn.net/xuanyuansan/article/details/41800721> 在机器视觉，图像处理领域，**卷积神经网络**取得了巨大的成功。本文将参考UF...

l\_better 2016-05-29 10:50:17 阅读数：1396

## 对LeNet-5卷积神经网络的理解

首先，我们给出**LeNet-5卷积神经网络**的整体框架：可以看出**LeNet-5**包含输入层共有8层，每一层都包含多个参数（权重）。C层代表的是卷积层，通过作，可以使原信号特征增强，并且降低噪音...

zhyj3038 2016-09-22 12:33:53 阅读数：791

## LeNet-5网络模型详解

深度学习交流QQ群：116270156 **卷积神经网络**是一种特殊的多层神经网络，像其它的神经网络一样，**卷积神经网络**也使用一种反向传播算法来进行训练，不同之处在于网络的结构。卷积神经...

sinat\_24143931 2018-01-03 10:55:20 阅读数：1104

## LeNet-5 经典卷积网络模型浅析

基本的一些卷积基础可以参考：CNN基础知识（2），这里不再介绍 1. **LeNet-5** 卷积模型 **LeNet-5** 卷积模型可以达到大约99.2%的正确率，**LeNet-5** 模型总共有7层 下...

lilong117194 2018-02-25 21:34:41 阅读数：987

## LeNet-5网络的参数及连接个数

原文：[http://vision.stanford.edu/cs598\\_spring07/papers/Lecun98.pdf](http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf) Gradient Based Learning Applied to ...

zyhzyh99 2016-12-10 16:18:07 阅读数：774

## 高效、准确、低成本的文字识别api

证件、文件一键扫描，准确率高达99%。经过海量用户和多种复杂场景考验



## LeNet-5 论文及原理分析(笨鸟角度)

#PS：要转载请注明出处，本人版权所有#PS:这个只是《我自己》理解，如果和你的 #原则相冲突，请谅解，勿喷Notes:本人此前只对opencv处理图像有一定的了解。前序：从毕业...

u011728480 2017-12-15 11:32:12 阅读数：146

## C++实现lenet-5 总结

最近自学了点神经网络深度学习，用C++写一个**lenet-5**做一个总结吧。**lenet-5**的原理，**卷积神经网络**的原理就不说了，大家自己到网上查，只总结一下我这个外行再实现**lenet-5**遇到的问题： ...

tangqinglin 2018-02-01 11:45:35 阅读数：236

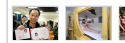
## 从LeNet-5看卷积神经网络CNNs

一、概述：自从2010年Hinton大神团队使用深度学习（Deep Learning）算法在ImageNet比赛中获得冠军之后，深度学习算法的触角在计算机视觉、语音识别、自然语言处理等领域不...

wangjian1204 2016-01-07 23:10:30 阅读数：9307



家用电梯



联系我们



关于 招聘  
©2018 CSDI  
百度提供

经营性网站  
网络110报警  
中国互联网  
北京互联网

## 深度卷积网络-LeNet-5深度解析

深度卷积网络 涉及问题： 1.每个图如何卷积： （1）一个图如何变成几个？ （2）卷积核如何选择？ 2.节点之间如何连接？ 3.S...

● ywxybzhd 2016-01-22 15:35:33 阅读数：761

## lenet-5,Alexnet详解以及tensorflow代码实现

Hinton的学生Alex Krizhevsky提出来深度卷积模型Alexnet，这是一种更深的更宽的版本。该模型在ILSVRS 2012年的比赛中一举夺冠，top-5错误的概率16.4%，识别...

● OliverkingLi 2017-06-28 17:35:08 阅读数：7737

## 免费云主机试用一年

云主机免费推荐吗

百度广告



## lenet-5训练MNIST

2015年03月12日 10.17MB

下载

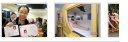
## 深度学习之一：卷积神经网络CNN经典模型整理（Lenet-5）

本文介绍以下几个CNN经典模型：Lenet（1986年）、Alexnet（2012年）、GoogleNet（2014年）、VGG（2014年）、Deep Residual Learning（2015

● liu\_zhlai 2016-11-02 01:08:08 阅读数：9066



家用电梯



联系我们



RA

关于 招聘  
©2018 CSDN  
百度提供

经营性网站  
网络110报警  
中国互联网  
北京互联网

个人资料



d5224

关注

原创  
11

粉丝  
29

喜欢  
12

评论  
14

等级： 博客 3

访问：8万+

积分：973

排名：5万+



在职研究生取消



最新文章

Caffe经典网络代码资料总结

matlab 选择文件夹、选择文件GUI和逐一读取文件夹中的文件

Caffe 学习：Eltwise层

深度学习中常用的调节参数

解读Batch Normalization

个人分类

CAFFE

18篇

深度学习

10篇



优化

ubuntu 64位 DrCom 32位 上网客户

cuda ubuntu nVidia驱动

展开

归档

2017年8月

2017年5月

2017年4月

2017年3月

2016年12月

展开

热门文章

ubuntu如何配置软件更新源和更新

阅读量：12811

卷积神经网络Lenet-5详解

阅读量：12513

Caffe 中 BN ( BatchNorm ) 层的参

值、方差和滑动系数解读

阅读量：7120

ubuntu14安装nVidia驱动和CUDA Toolkit的

安装和调试

阅读量：6994

机器学习(Machine Learning)&深度学习(De

ep Learning)资料集合

阅读量：6658

最新评论

win7 x64系统配置caffe...

zhou578325336：这方法根本就不行，没事别瞎

转。

简单4步在win7 X64+cud...

zhxhappy0925：楼主，装cuda的时候，显卡一定

要是N卡吗？

机器学习(Machine Lear...

Jessie\_804：谢谢分享

简单4步在win7 X64+cud...


d5224：[reply]zn\_qq[/reply] 链接：http://pan.baid

u.com/s/1k...


简单4步在win7 X64+cud...

d5224：[reply]qq\_35463898[/reply] 需要编译的，


不过很简单



家用电梯




联系我们



关于 招聘

©2018 CSDN

 百度提供

经营性网站备案

网络110报警

中国互联网斗

北京互联网通