

Charlotte77

数学系的数据挖掘民工(公众号:CharlotteDataMining, 深度学习技术交流qq群:339120614)最新深度学习免费学习视频请移步我的B站: <https://www.bilibili.com/video/av75414647>

博客园 首页 新随笔 联系 管理 订阅

随笔- 56 文章- 0 评论- 1525

【深度学习系列】卷积神经网络CNN原理详解(一)——基本原理

上篇文章我们给出了用paddlepaddle来做手写数字识别的示例, 并对网络结构进行了调整, 提高了识别的精度。有的同学表示不是很理解原理, 为什么传统的机器学习算法, 简单的神经网络(如多层感知机)都可以识别手写数字, 我们要采用卷积神经网络CNN来进行识别呢? CNN到底是怎么识别的? 用CNN有哪些优势呢? 我们下面就来简单分析一下。在讲CNN之前, 为避免完全零基础的人看不懂后面的讲解, 我们先简单回顾一下传统的神经网络的基本知识。

神经网络的预备知识

为什么要用神经网络?

- 特征提取的高效性。

大家可能会疑惑, 对于同一个分类任务, 我们可以用机器学习的算法来做, 为什么要用神经网络呢? 大家回顾一下, 一个分类任务, 我们在用机器学习算法来做时, 首先要明确feature和label, 然后把这个数据"灌"到算法里去训练, 最后保存模型, 再来预测分类的准确性。但是这就有个问题, 即我们需要实现确定好特征, 每一个特征即为一个维度, 特征数目过少, 我们可能无法精确的分类出来, 即我们所说的欠拟合, 如果特征数目过多, 可能会导致我们在分类过程中过于注重某个特征导致分类错误, 即过拟合。

举个简单的例子, 现在有一堆数据集, 让我们分类出西瓜和冬瓜, 如果只有两个特征: 形状和颜色, 可能没法区分来; 如果特征的维度有: 形状、颜色、瓜瓢颜色、瓜皮的花纹等等, 可能很容易分类出来; 如果我们的特征是: 形状、颜色、瓜瓢颜色、瓜皮花纹、瓜蒂、瓜籽的数量, 瓜籽的颜色、瓜籽的大小、瓜籽的分布情况、瓜籽的XXX等等, 很有可能会过拟合, 譬如有的冬瓜的瓜籽数量和西瓜的类似, 模型训练后这类特征的权重较高, 就很容易错分。这就导致我们在特征工程上需要花很多时间和精力, 才能使模型训练得到一个很好的效果。然而神经网络的出现使我们不需要做大量的特征工程, 譬如提前设计好特征的内容或者说特征的数量等等, 我们可以直接把数据灌进去, 让它自己训练, 自我"修正", 即可得到一个较好的效果。

- 数据格式的简易性

在一个传统的机器学习分类问题中, 我们"灌"进去的数据是不能直接灌进去的, 需要对数据进行一些处理, 譬如量纲的归一化, 格式的转化等等, 不过在神经网络里我们不需要额外的对数据做过多的处理, 具体原因可以看后面的详细推导。

- 参数数目的少量性

在面对一个分类问题时, 如果用SVM来做, 我们需要调整的参数需要调整核函数, 惩罚因子, 松弛变量等等, 不同的参数组合对于模型的效果也不一样, 想要迅速而又准确的调到最适合模型的参数需要对背后理论知识的深入了解(当然, 如果你说全部都试一遍也是可以的, 但是花的时间可能会更多), 对于一个基本的三层神经网络来说(输入-隐含-输出), 我们只需要初始化时给每一个神经元上随机的赋予一个权重 w 和偏置项 b , 在训练过程中, 这两个参数会不断的修正, 调整到最优质, 使模型的误差最小。所以从这个角度来看, 我

本博客所有内容以学习、研究和分享为主, 如需转载, 请联系本人, 标明作者和出处, 并且是非商业用途, 谢谢!

Email: charlotte77_hu@sina.com

Github: <https://github.com/huxiaoman7>

知乎: https://www.zhihu.com/people/charlotte77_hu

微博: <http://weibo.com/2189505447/profile?topnav=1&wvr=6>

微信公众号: Charlotte数据挖掘



昵称: Charlotte77

园龄: 4年8个月

荣誉: 推荐博客

粉丝: 3743

关注: 8

+加关注

2020年8月						
<	日	一	二	三	四	五
	26	27	28	29	30	31
	2	3	4	5	6	7
	9	10	11	12	13	14
	16	17	18	19	20	21
	23	24	25	26	27	28
	30	31	1	2	3	4

搜索

<input type="text"/>	找找看
<input type="text"/>	谷歌搜索

常用链接

[我的随笔](#)

们对于调参的背后理论知识并不需要过于精通(只不过做多了之后可能会有一些经验,在初始值时赋予的值更科学,收敛的更快罢了)

有哪些应用？

应用非常广,不过大家注意一点,我们现在所说的神经网络,并不能称之为深度学习,神经网络很早就出现了,只不过现在因为不断的加深了网络层,复杂化了网络结构,才成为深度学习,并在图像识别、图像检测、语音识别等方面取得了不错的效果。

基本网络结构

一个神经网络最简单的结构包括输入层、隐含层和输出层,每一层网络有多个神经元,上一层的神经元通过激活函数映射到下一层神经元,每个神经元之间有相对应的权值,输出即为我们的分类类别。

详细数学推导

去年中旬我参考吴恩达的UFLDL和mattmazur的博客写了篇文章详细讲解了一个最简单的神经网络从前向传播到反向传播的直观推导,大家可以先看看这篇文章--[一文看懂神经网络中的反向传播法--BackPropagation](#)。

优缺点

前面说了很多优点,这里就不多说了,简单说说缺点吧。我们试想一下如果加深我们的网络层,每一个网络层增加神经元的数量,那么参数的个数将是 $M*N$ (m 为网络层数, N 为每层神经元个数),所需的参数会非常多,参数一多,模型就复杂了,越是复杂的模型就越不好调参,也更容易过拟合。此外我们从神经网络的反向传播的过程来看,梯度在反向传播时,不断的迭代会导致梯度越来越小,即梯度消失的情况,梯度一旦趋于0,那么权值就无法更新,这个神经元相当于是不起作用了,也就很难导致收敛。尤其是在图像领域,用最基本的神经网络,是不太合适的。后面我们会详细讲讲为啥不合适。

为什么要用卷积神经网络？

传统神经网络的劣势

前面说到在图像领域,用传统的神经网络并不合适。我们知道,图像是由一个个像素点构成,每个像素点有三个通道,分别代表RGB颜色,那么,如果一个图像的尺寸是(28, 28, 1),即代表这个图像的是一个长宽均为28, channel为1的图像(channel也叫depth,此处1代表灰色图像)。如果使用全连接的网络结构,即,网络中的神经与与相邻层上的每个神经元均连接,那就意味着我们的网络有 $28 * 28 = 784$ 个神经元,hidden层采用了15个神经元,那么简单计算一下,我们需要的参数个数(w 和 b)就有: $784 * 15 * 10 + 15 * 10 = 117625$ 个,这个参数太多了,随便进行一次反向传播计算量都是巨大的,从计算资源和调参的角度都不建议用传统的神经网络。(评论中有同学对这个参数计算不太理解,我简单说一下:图片是由像素点组成的,用矩阵表示的, $28 * 28$ 的矩阵,肯定是没法直接放到神经元里的,我们得把它"拍平",变成一个 $28 * 28 = 784$ 的一列向量,这一列向量和隐含层的15个神经元连接,就有 $784 * 15 = 11760$ 个权重 w , 隐含层和最后的输出层的10个神经元连接,就有 $784 * 15 = 117600$ 个权重 w , 再加上隐含层的偏置项15个和输出层的偏置项10个,就是:117625个参数了)

我的评论
我的参与
最新评论
我的标签

最新随笔

1. 谈谈坚持这件小事
2. 我在北京这几年 (全)
3. 【原】深度学习的一些经验总结和建议 | To do v.s Not To Do
4. 如何高效利用一场技术分享？
5. 深度学习分布式训练及CTR预估模型应用
6. 两个月刷完Leetcode前400题经验总结
7. 【机器学习】如何解决数据不平衡问题
8. LeetCode刷题专栏第一篇--思维导图&时间安排
9. 【资料总结】| Deep Reinforcement Learning 深度强化学习
10. 2018年总结与2019年目标与计划

我的标签

深度学习(22)
机器学习(10)
数据挖掘(5)
Spark(4)
学习心得(3)
数据挖掘(2)
推荐系统(2)
文本挖掘(2)
LeetCode(2)
年度总结(2)
更多

积分与排名

积分 - 177025
排名 - 3422

随笔分类 (56)

Spark(7)
机器学习笔记(12)
深度学习(23)
数据挖掘(9)
推荐系统(2)
文本挖掘(3)

随笔档案 (56)

2019年8月(2)
2019年7月(2)
2019年5月(2)
2019年3月(1)
2019年2月(1)
2019年1月(2)
2018年6月(1)
2018年5月(1)
2018年3月(1)
2018年2月(2)
2018年1月(4)
2017年12月(4)
2017年11月(4)
2017年10月(2)
2017年9月(1)
2016年12月(1)

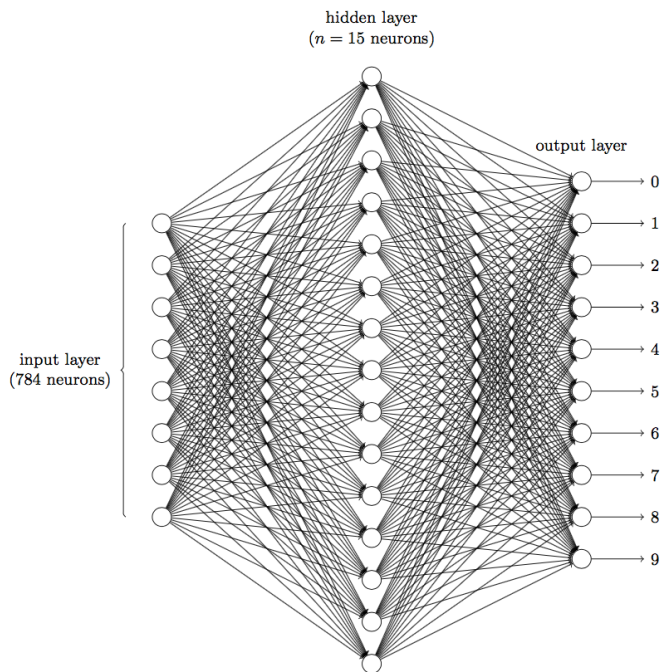


图1 三层神经网络

别手写数字

卷积神经网络是什么？

三个基本层

• 卷积层 (Convolutional Layer)

上文提到我们用传统的三层神经网络需要大量的参数，原因在于每个神经元都和相邻层的神经元相连接，但是思考一下，这种连接方式是必须的吗？全连接层的方式对于图像数据来说似乎显得不这么友好，**因为图像本身具有“二维空间特征”，通俗点说就是局部特性**。譬如我们看一张猫的图片，可能看到猫的眼镜或者嘴巴就知道这是张猫片，而不需要说每个部分都看完了才知道，啊，原来这个是猫啊。所以如果我们用某种方式对一张图片的某个典型特征识别，那么这张图片的类别也就知道了。这个时候就产生了卷积的概念。举个例子，现在有一个4*4的图像，我们设计两个卷积核，看看运用卷积核后图片会变成什么样。

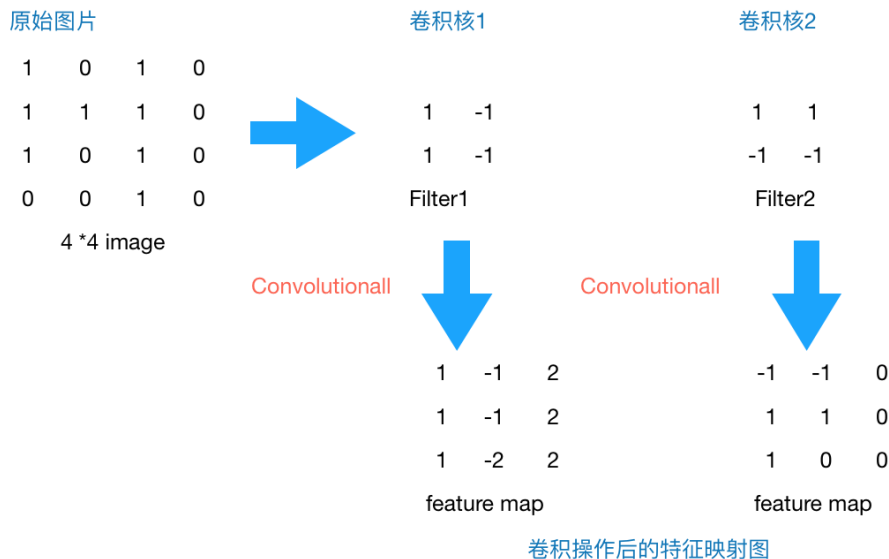


图2 4*4 image与两个2*2的卷积核操作结果

2016年7月(3)
2016年6月(3)
2016年5月(9)
2016年4月(6)
2016年3月(1)
2015年12月(3)

最新评论

1. Re:谈谈坚持这件小事
感谢，你已经帮助到我了。

--外方

2. Re:一文看懂神经网络中的反向传播法——BackProp
agation
隐藏层的误差是等于总误差，还是等于E_1w5+E_2w6
啊

--Benys

3. Re:【深度学习系列】卷积神经网络详解(二)——自己
手写一个卷积神经网络
@风中de石头 谢谢，已经能够运行啦...

--3079779149

4. Re:我在北京这几年 (全)
很幸运可以看到楼主的文章，关注楼主。大学的第一个暑假，因为疫情待在家好几个月，想在暑假学习深度学习相关知识却因为种种困难犹豫不前，效率低下，希望自己可以坚持下去，为了自己的目标，为了成为更好的自己。..

--ywqa

5. Re:我在北京这几年 (全)
博主的文章很真实，比那些上来就鼓吹理想奋斗的文章的感触更深，能够更理性更客观的认识程序员这个职业，以及在北京一步一步奋斗所经历的点点滴滴。我也要冲向北京了，争取早日秋招上岸！！

--MIIeO

6. Re:2018年总结与2019年目标与计划
这个复盘好，学习了

--6671

7. Re:一文看懂神经网络中的反向传播法——BackProp
agation
Charlotte you are so great Refer to there is back
propagation.py that I modified will help you a lot
...

--fatalfeel

8. Re:我在北京这几年 (全)
姐姐写的很好

--Details_K

9. Re:【深度学习系列】卷积神经网络CNN原理详解(一)
)——基本原理
您好博主，为啥我感觉一开始最上面那个，参数个数是这样算的：784+(15784+151)+(1015+101)，您 784 1510，乘以10没有看懂是为什么，w^[l]的维数不是 (n^[l], n...

--douzujun

10. Re:【深度学习系列】卷积神经网络详解(二)——自己
手写一个卷积神经网络
我没看出来有偏置项了呀？

--小艾1

阅读排行榜

1. 一文看懂神经网络中的反向传播法——BackPropaga
tion(288897)

由上图可以看到，原始图片是一张灰度图片,每个位置表示的是像素值，0表示白色，1表示黑色，（0，1）区间的数值表示灰色。对于这个4*4的图像，我们采用两个2*2的卷积核来计算。设定步长为1，即每次以2*2的固定窗口往右滑动一个单位。以第一个卷积核filter1为例，计算过程如下：

```
1 feature_map1(1,1) = 1*1 + 0*(-1) + 1*1 + 1*(-1) = 1
2 feature_map1(1,2) = 0*1 + 1*(-1) + 1*1 + 1*(-1) = -1
3 ...
4 feature_map1(3,3) = 1*1 + 0*(-1) + 1*1 + 0*(-1) = 2
```

可以看到这就是最简单的内积公式。`feature_map1(1,1)`表示在通过第一个卷积核计算完后得到的`feature_map`的第一行第一列的值，随着卷积核的窗口不断的滑动，我们可以计算出一个3*3的`feature_map1`;同理可以计算通过第二个卷积核进行卷积运算后的`feature_map2`，那么这一层卷积操作就完成了。`feature_map`尺寸计算公式：`[(原图片尺寸 - 卷积核尺寸) / 步长] + 1`。这一层我们设定了两个2*2的卷积核，在paddlepaddle里是这样定义的：

```
1 conv_pool_1 = paddle.networks.simple_img_conv_pool(
2     input=img,
3     filter_size=3,
4     num_filters=2,
5     num_channel=1,
6     pool_stride=1,
7     act=paddle.activation.Relu())
```

这里调用了`networks`里`simple_img_conv_pool`函数，激活函数是`Relu`(修正线性单元)，我们来看一看源码里外层接口是如何定义的：

```
1 def simple_img_conv_pool(input,
2     filter_size,
3     num_filters,
4     pool_size,
5     name=None,
6     pool_type=None,
7     act=None,
8     groups=1,
9     conv_stride=1,
10    conv_padding=0,
11    bias_attr=None,
12    num_channel=None,
13    param_attr=None,
14    shared_bias=True,
15    conv_layer_attr=None,
16    pool_stride=1,
17    pool_padding=0,
18    pool_layer_attr=None):
19    """
20    Simple image convolution and pooling group.
21    Img input => Conv => Pooling => Output.
22    :param name: group name.
23    :type name: basestring
24    :param input: input layer.
25    :type input: LayerOutput
26    :param filter_size: see img_conv_layer for details.
27    :type filter_size: int
28    :param num_filters: see img_conv_layer for details.
```

2. 【深度学习系列】卷积神经网络CNN原理详解(一)——基本原理(178639)
3. 三个月教你从零入门深度学习(59938)
4. 【深度学习系列】卷积神经网络详解(二)——自己手写一个卷积神经网络(53580)
5. 机器学习基础与实践（一）----数据清洗(52731)
6. 如何用卷积神经网络CNN识别手写数字集？(33461)
7. 机器学习基础与实践（二）----数据转换(31268)
8. 用Tensorflow让神经网络自动创造音乐(27092)
9. 【深度学习Deep Learning】资料大全(25453)
10. 【原】数据分析/数据挖掘/机器学习---- 必读书目(23376)

评论排行榜

1. 三个月教你从零入门深度学习(219)
2. 我在北京这几年（全）(161)
3. 一文看懂神经网络中的反向传播法——BackPropagation(154)
4. 2015年总结与2016年目标和计划(125)
5. 【深度学习系列】卷积神经网络CNN原理详解(一)——基本原理(108)
6. 2018年总结与2019年目标与计划(88)
7. 2017年总结与2018年目标和计划(72)
8. 【深度学习系列】卷积神经网络详解(二)——自己手写一个卷积神经网络(71)
9. 坑爹的2016年总结(57)
10. 两个月刷完Leetcode前400题经验总结(33)

推荐排行榜

1. 三个月教你从零入门深度学习(238)
2. 一文看懂神经网络中的反向传播法——BackPropagation(159)
3. 我在北京这几年（全）(97)
4. 【深度学习系列】卷积神经网络CNN原理详解(一)——基本原理(96)
5. 【深度学习系列】卷积神经网络详解(二)——自己手写一个卷积神经网络(51)


```
29     :type num_filters: int
30     :param pool_size: see img_pool_layer for details.
31     :type pool_size: int
32     :param pool_type: see img_pool_layer for details.
33     :type pool_type: BasePoolingType
34     :param act: see img_conv_layer for details.
35     :type act: BaseActivation
36     :param groups: see img_conv_layer for details.
37     :type groups: int
38     :param conv_stride: see img_conv_layer for details.
39     :type conv_stride: int
40     :param conv_padding: see img_conv_layer for details.
41     :type conv_padding: int
42     :param bias_attr: see img_conv_layer for details.
43     :type bias_attr: ParameterAttribute
44     :param num_channel: see img_conv_layer for details.
45     :type num_channel: int
46     :param param_attr: see img_conv_layer for details.
47     :type param_attr: ParameterAttribute
48     :param shared_bias: see img_conv_layer for details.
49     :type shared_bias: bool
50     :param conv_layer_attr: see img_conv_layer for details.
51     :type conv_layer_attr: ExtraLayerAttribute
52     :param pool_stride: see img_pool_layer for details.
53     :type pool_stride: int
54     :param pool_padding: see img_pool_layer for details.
55     :type pool_padding: int
56     :param pool_layer_attr: see img_pool_layer for details.
57     :type pool_layer_attr: ExtraLayerAttribute
58     :return: layer's output
59     :rtype: LayerOutput
60     """
61     _conv_ = img_conv_layer(
62         name="%s_conv" % name,
63         input=input,
64         filter_size=filter_size,
65         num_filters=num_filters,
66         num_channels=num_channel,
67         act=act,
68         groups=groups,
69         stride=conv_stride,
70         padding=conv_padding,
71         bias_attr=bias_attr,
72         param_attr=param_attr,
73         shared_biases=shared_bias,
74         layer_attr=conv_layer_attr)
75     return img_pool_layer(
76         name="%s_pool" % name,
77         input=_conv_,
78         pool_size=pool_size,
79         pool_type=pool_type,
80         stride=pool_stride,
81         padding=pool_padding,
82         layer_attr=pool_layer_attr)
```



我们在[Paddle/python/paddle/v2/framework/nets.py](https://paddle.pytorch/paddle/v2/framework/nets.py)里可以看到
simple_img_conv_pool这个函数的定义：

```
1 def simple_img_conv_pool(input,
2                           num_filters,
3                           filter_size,
4                           pool_size,
5                           pool_stride,
6                           act,
```



```
7         pool_type='max',
8         main_program=None,
9         startup_program=None):
10     conv_out = layers.conv2d(
11         input=input,
12         num_filters=num_filters,
13         filter_size=filter_size,
14         act=act,
15         main_program=main_program,
16         startup_program=startup_program)
17
18     pool_out = layers.pool2d(
19         input=conv_out,
20         pool_size=pool_size,
21         pool_type=pool_type,
22         pool_stride=pool_stride,
23         main_program=main_program,
24         startup_program=startup_program)
25     return pool_out
```



可以看到这里面有两个输出，conv_out是卷积输出值，pool_out是池化输出值，最后只返回池化输出的值。conv_out和pool_out分别又调用了layers.py的conv2d和pool2d，去layers.py里我们可以看到conv2d和pool2d是如何实现的：

conv2d:



```
def conv2d(input,
            num_filters,
            name=None,
            filter_size=[1, 1],
            act=None,
            groups=None,
            stride=[1, 1],
            padding=None,
            bias_attr=None,
            param_attr=None,
            main_program=None,
            startup_program=None):
    helper = LayerHelper('conv2d', **locals())
    dtype = helper.input_dtype()

    num_channels = input.shape[1]
    if groups is None:
        num_filter_channels = num_channels
    else:
        if num_channels % groups is not 0:
            raise ValueError("num_channels must be divisible by groups.")
        num_filter_channels = num_channels / groups

    if isinstance(filter_size, int):
        filter_size = [filter_size, filter_size]
    if isinstance(stride, int):
        stride = [stride, stride]
    if isinstance(padding, int):
        padding = [padding, padding]

    input_shape = input.shape
    filter_shape = [num_filters, num_filter_channels] + filter_size

    std = (2.0 / (filter_size[0]**2 * num_channels))**0.5
    filter = helper.create_parameter(
        attr=helper.param_attr,
        shape=filter_shape,
```

```
dtype=dtype,
    initializer=NormalInitializer(0.0, std, 0))
pre_bias = helper.create_tmp_variable(dtype)

helper.append_op(
    type='conv2d',
    inputs={
        'Input': input,
        'Filter': filter,
    },
    outputs={"Output": pre_bias},
    attrs={'strides': stride,
          'paddings': padding,
          'groups': groups})

pre_act = helper.append_bias_op(pre_bias, 1)

return helper.append_activation(pre_act)
```



pool2d:



```
1 def pool2d(input,
2             pool_size,
3             pool_type,
4             pool_stride=[1, 1],
5             pool_padding=[0, 0],
6             global_pooling=False,
7             main_program=None,
8             startup_program=None):
9     if pool_type not in ["max", "avg"]:
10         raise ValueError(
11             "Unknown pool_type: '%s'. It can only be 'max' or 'avg'."
12             str(pool_type))
13     if isinstance(pool_size, int):
14         pool_size = [pool_size, pool_size]
15     if isinstance(pool_stride, int):
16         pool_stride = [pool_stride, pool_stride]
17     if isinstance(pool_padding, int):
18         pool_padding = [pool_padding, pool_padding]
19
20     helper = LayerHelper('pool2d', **locals())
21     dtype = helper.input_dtype()
22     pool_out = helper.create_tmp_variable(dtype)
23
24     helper.append_op(
25         type="pool2d",
26         inputs={"X": input},
27         outputs={"Out": pool_out},
28         attrs={
29             "poolingType": pool_type,
30             "ksize": pool_size,
31             "globalPooling": global_pooling,
32             "strides": pool_stride,
33             "paddings": pool_padding
34         })
35
36     return pool_out
```



大家可以看到，具体的实现方式还调用了[layers_helper.py](#):



```
1 import copy
2 import itertools
3
4 from paddle.v2.framework.framework import Variable, g_main_program, \
5     g_startup_program, unique_name, Program
6 from paddle.v2.framework.initializer import ConstantInitializer, \
7     UniformInitializer
8
9
10 class LayerHelper(object):
11     def __init__(self, layer_type, **kwargs):
12         self.kwargs = kwargs
13         self.layer_type = layer_type
14         name = self.kwargs.get('name', None)
15         if name is None:
16             self.kwargs['name'] = unique_name(self.layer_type)
17
18     @property
19     def name(self):
20         return self.kwargs['name']
21
22     @property
23     def main_program(self):
24         prog = self.kwargs.get('main_program', None)
25         if prog is None:
26             return g_main_program
27         else:
28             return prog
29
30     @property
31     def startup_program(self):
32         prog = self.kwargs.get('startup_program', None)
33         if prog is None:
34             return g_startup_program
35         else:
36             return prog
37
38     def append_op(self, *args, **kwargs):
39         return self.main_program.current_block().append_op(*args, **kwargs)
40
41     def multiple_input(self, input_param_name='input'):
42         inputs = self.kwargs.get(input_param_name, [])
43         type_error = TypeError(
44             "Input of {0} layer should be Variable or sequence of Variable".
45             format(self.layer_type))
46         if isinstance(inputs, Variable):
47             inputs = [inputs]
48         elif not isinstance(inputs, list) and not isinstance(inputs, tuple):
49             raise type_error
50         else:
51             for each in inputs:
52                 if not isinstance(each, Variable):
53                     raise type_error
54         return inputs
55
56     def input(self, input_param_name='input'):
57         inputs = self.multiple_input(input_param_name)
58         if len(inputs) != 1:
59             raise "{0} layer only takes one input".format(self.layer_type)
60         return inputs[0]
61
62     @property
63     def param_attr(self):
64         default = {'name': None, 'initializer': UniformInitializer()}
65         actual = self.kwargs.get('param_attr', None)
66         if actual is None:
67             actual = default
68         for default_field in default.keys():
69             if default_field not in actual:
```



```

70         actual[default_field] = default[default_field]
71     return actual
72
73     def bias_attr(self):
74         default = {'name': None, 'initializer': ConstantInitializer()}
75         bias_attr = self.kwargs.get('bias_attr', None)
76         if bias_attr is True:
77             bias_attr = default
78
79         if isinstance(bias_attr, dict):
80             for default_field in default.keys():
81                 if default_field not in bias_attr:
82                     bias_attr[default_field] = default[default_field]
83         return bias_attr
84
85     def multiple_param_attr(self, length):
86         param_attr = self.param_attr
87         if isinstance(param_attr, dict):
88             param_attr = [param_attr]
89
90         if len(param_attr) != 1 and len(param_attr) != length:
91             raise ValueError("parameter number mismatch")
92         elif len(param_attr) == 1 and length != 1:
93             tmp = [None] * length
94             for i in xrange(length):
95                 tmp[i] = copy.deepcopy(param_attr[0])
96             param_attr = tmp
97         return param_attr
98
99     def iter_inputs_and_params(self, input_param_name='input'):
100         inputs = self.multiple_input(input_param_name)
101         param_attrs = self.multiple_param_attr(len(inputs))
102         for ipt, param_attr in itertools.izip(inputs, param_attrs):
103             yield ipt, param_attr
104
105     def input_dtype(self, input_param_name='input'):
106         inputs = self.multiple_input(input_param_name)
107         dtype = None
108         for each in inputs:
109             if dtype is None:
110                 dtype = each.data_type
111             elif dtype != each.data_type:
112                 raise ValueError("Data Type mismatch")
113         return dtype
114
115     def create_parameter(self, attr, shape, dtype, suffix='w',
116                         initializer=None):
117         # Deepcopy the attr so that parameters can be shared in program
118         attr_copy = copy.deepcopy(attr)
119         if initializer is not None:
120             attr_copy['initializer'] = initializer
121         if attr_copy['name'] is None:
122             attr_copy['name'] = unique_name(".".join([self.name, suffix]))
123         self.startup_program.global_block().create_parameter(
124             dtype=dtype, shape=shape, **attr_copy)
125         return self.main_program.global_block().create_parameter(
126             name=attr_copy['name'], dtype=dtype, shape=shape)
127
128     def create_tmp_variable(self, dtype):
129         return self.main_program.current_block().create_var(
130             name=unique_name(".".join([self.name, 'tmp'])),
131             dtype=dtype,
132             persistable=False)
133
134     def create_variable(self, *args, **kwargs):
135         return self.main_program.current_block().create_var(*args, **kwargs)
136
137     def create_global_variable(self, persistable=False, *args, **kwargs):
138         return self.main_program.global_block().create_var(

```

```

139         *args, persistable=persistable, **kwargs)
140
141     def set_variable_initializer(self, var, initializer):
142         assert isinstance(var, Variable)
143         self.startup_program.global_block().create_var(
144             name=var.name,
145             type=var.type,
146             dtype=var.data_type,
147             shape=var.shape,
148             persistable=True,
149             initializer=initializer)
150
151     def append_bias_op(self, input_var, num_flatten_dims=None):
152         """
153         Append bias operator and return its output. If the user does not set
154
155         bias_attr, append_bias_op will return input_var
156
157         :param input_var: the input variable. The len(input_var.shape) is la
158         rger
159         or equal than 2.
160         :param num_flatten_dims: The input tensor will be flatten as a matri
161         x
162         when adding bias.
163         `matrix.shape = product(input_var.shape[0:num_flatten_dims]), produc
164         t(
165             input_var.shape[num_flatten_dims:])`
166         """
167         if num_flatten_dims is None:
168             num_flatten_dims = self.kwargs.get('num_flatten_dims', None)
169             if num_flatten_dims is None:
170                 num_flatten_dims = 1
171
172         size = list(input_var.shape[num_flatten_dims:])
173         bias_attr = self.bias_attr()
174         if not bias_attr:
175             return input_var
176
177         b = self.create_parameter(
178             attr=bias_attr, shape=size, dtype=input_var.data_type, suffix='b
179         ')
180
181         tmp = self.create_tmp_variable(dtype=input_var.data_type)
182         self.append_op(
183             type='elementwise_add',
184             inputs={'X': [input_var],
185                    'Y': [b]},
186             outputs={'Out': [tmp]})
187         return tmp
188
189     def append_activation(self, input_var):
190         act = self.kwargs.get('act', None)
191         if act is None:
192             return input_var
193         if isinstance(act, basestring):
194             act = {'type': act}
195         tmp = self.create_tmp_variable(dtype=input_var.data_type)
196         act_type = act.pop('type')
197         self.append_op(
198             type=act_type,
199             inputs={"X": [input_var]},
200             outputs={"Y": [tmp]},
201             attrs=act)
202         return tmp

```



详细的源码细节我们下一节会讲这里指写一下实现的方式和调用的函数。

所以这个卷积过程就完成了。从上文的计算中我们可以看到，**同一层的神经元可以共享卷积核**，那么对于高位数据的处理将会变得非常简单。并且使用卷积核后图片的尺寸变小，方便后续计算，并且**我们不需要手动去选取特征，只用设计好卷积核的尺寸，数量和滑动的步长就可以让它自己去训练了**，省时又省力啊。

为什么卷积核有效？

那么问题来了，虽然我们知道了卷积核是如何计算的，但是为什么使用卷积核计算后分类效果要优于普通的神经网络呢？我们仔细来看一下上面计算的结果。通过第一个卷积核计算后的feature_map是一个三维数据，在第三列的绝对值最大，说明原始图片上对应的地方有一条垂直方向的特征，即像素数值变化较大；而通过第二个卷积核计算后，第三列的数值为0，第二行的数值绝对值最大，说明原始图片上对应的地方有一条水平方向的特征。

仔细思考一下，这个时候，我们设计的两个卷积核分别能够提取，或者说检测出原始图片的特定的特征。此时**我们其实就可以把卷积核就理解为特征提取器**啊！现在就明白了，为什么我们只需要把图片数据灌进去，设计好卷积核的尺寸、数量和滑动的步长就可以让自动提取出图片的某些特征，从而达到分类的效果啊！

注：1.此处的卷积运算是两个卷积核大小的矩阵的内积运算，不是矩阵乘法。即相同位置的数字相乘再相加求和。不要弄混淆了。

2.卷积核的公式有很多，这只是最简单的一种。我们所说的卷积核在数字信号处理里也叫滤波器，那滤波器的种类就多了，均值滤波器，高斯滤波器，拉普拉斯滤波器等，不过，不管是什么滤波器，都只是一种数学运算，无非就是计算更复杂一点。

3.每一层的卷积核大小和个数可以自己定义，不过**一般情况下，根据实验得到的经验来看，会在越靠近输入层的卷积层设定少量的卷积核，越往后，卷积层设定的卷积核数目就越多**。具体原因大家可以先思考一下，小结里会解释原因。

池化层 (Pooling Layer)

通过上一层2*2的卷积核操作后，我们将原始图像由4*4的尺寸变为了3*3的一个新的图片。池化层的主要目的是通过降采样的方式，在不影响图像质量的情况下，压缩图片，减少参数。简单来说，假设现在设定池化层采用**MaxPooling**，大小为2*2，步长为1，取每个窗口最大的数值重新，那么图片的尺寸就会由3*3变为2*2： $(3-2)+1=2$ 。从上例来看，会有如下变换：

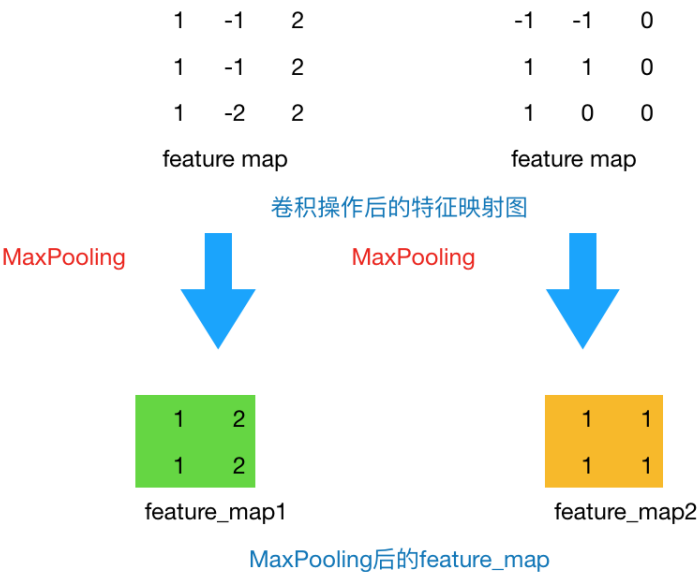


图3 Max Pooling结果

通常来说，池化方法一般有以下两种：

- **MaxPooling**：取滑动窗口里最大的值
- **AveragePooling**：取滑动窗口内所有值的平均值

为什么采用Max Pooling？

从计算方式来看，算是最简单的一种了，取max即可，但是这也引发一个思考，为什么需要Max Pooling，意义在哪里？如果我们只取最大值，那其他的值被舍弃难道就没有影响吗？不会损失这部分信息吗？如果认为这些信息是可损失的，那么是否意味着我们在进行卷积操作后仍然产生了一些不必要的冗余信息呢？

其实从上文分析卷积核为什么有效的原因来看，每一个卷积核可以看做一个特征提取器，不同的卷积核负责提取不同的特征，我们例子中设计的第一个卷积核能够提取出“垂直”方向的特征，第二个卷积核能够提取出“水平”方向的特征，那么我们对其进行Max Pooling操作后，提取出的是真正能够识别特征的数值，其余被舍弃的数值，对于我提取特定的特征并没有特别大的帮助。那么在进行后续计算使，减小了feature map的尺寸，从而减少参数，达到减小计算量，却不损失效果的情况。

不过并不是所有情况Max Pooling的效果都很好，有时候有些周边信息也会对某个特定特征的识别产生一定效果，那么这个时候舍弃这部分“不重要”的信息，就不划算了。所以具体情况得具体分析，如果加了Max Pooling后效果反而变差了，不如把卷积后不加Max Pooling的结果与卷积后加了Max Pooling的结果输出对比一下，看看Max Pooling是否对卷积核提取特征起了反效果。

Zero Padding

所以到现在为止，我们的图片由4*4，通过卷积层变为3*3，再通过池化层变化2*2，如果我们再添加层，那么图片岂不是会越变越小？这个时候我们会引出“Zero Padding”（补零），它可以帮助我们保证每次经过卷积或池化输出后图片的大小不变，如，上述例子我们如果加入Zero Padding，再采用3*3的卷积核，那么变换后的图片尺寸与原图片尺寸相同，如下图所示：

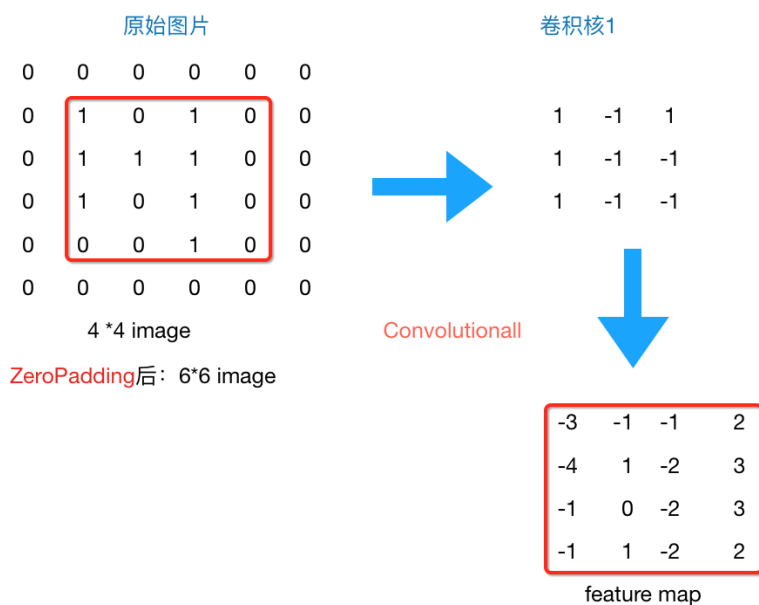


图4 zero padding结果

通常情况下，我们希望图片做完卷积操作后保持图片大小不变，所以我们一般会选择尺寸为3*3的卷积核和1的zero padding，或者5*5的卷积核与2的zero padding，这样通过计算后，可以保留图片的原始尺寸。那么加入zero padding后的feature_map尺寸 = (width + 2 * padding_size - filter_size) / stride + 1

注：这里的width也可换成height，此处是默认正方形的卷积核，weight = height，

如果两者不相等，可以分开计算，分别补零。

• Flatten层 & Fully Connected Layer

到这一步，其实我们的一个完整的“卷积部分”就算完成了，如果想要叠加层数，一般也是叠加“Conv-MaxPooling”，通过不断的设计卷积核的尺寸，数量，提取更多的特征，最后识别不同类别的物体。做完Max Pooling后，我们就会把这些数据“拍平”，丢到Flatten层，然后把Flatten层的output放到full connected Layer里，采用softmax对其进行分类。

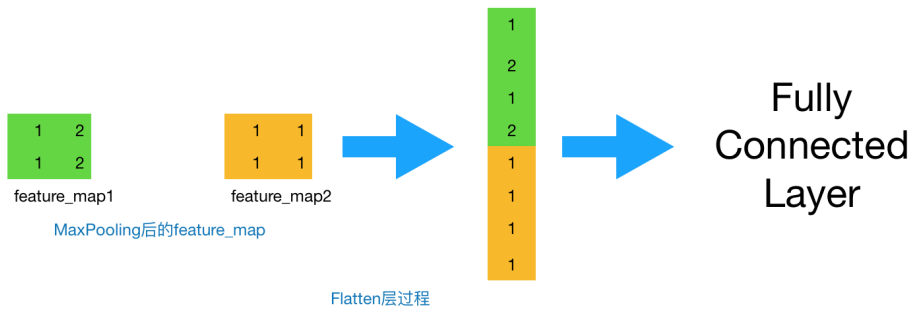


图5 Flatten过程





• 小结

这一节我们介绍了最基本的卷积神经网络的基本层的定义，计算方式和起的作用。有几个小问题可以供大家思考一下：

- 1.卷积核的尺寸必须为正方形吗？可以为长方形吗？如果是长方形应该怎么计算？
- 2.卷积核的个数如何确定？每一层的卷积核的个数都是相同的吗？
- 3.步长的向右和向下移动的幅度必须是一样的吗？

如果对上述的讲解真的弄懂了的话，其实这几个问题并不难回答。下面给出我的想法，可以作为参考：


1.卷积核的尺寸不一定非得为正方形。长方形也可以，只不过通常情况下为正方形。如果要设置为长方形，那么首先得保证这层的输出形状是整数，不能是小数。如果你的图像是边长为 28 的正方形。那么卷积层的输出就满足 $\lfloor (28 - \text{kernel_size}) / \text{stride} \rfloor + 1$ ，这个数值得是整数才行，否则没有物理意义。譬如，你算得一个边长为 3.6 的 feature map 是没有物理意义的。pooling 层同理。FC 层的输出形状总是满足整数，其唯一的要求就是整个训练过程中 FC 层的输入得是定长的。如果你的图像不是正方形。那么在制作数据时，可以缩放到统一大小（非正方形），再使用非正方形的 kernel_size 来使得卷积层的输出依然是整数。总之，撇开网络结果设定的好坏不谈，其本质上就是在做算术应用题：如何使得各层的输出是整数。

2.由经验确定。通常情况下，靠近输入的卷积层，譬如第一层卷积层，会找出一些共性的特征，如手写数字识别中第一层我们设定卷积核个数为5个，一般是找出诸如“横线”、“竖线”、“斜线”等共性特征，我们称之为basic feature，经过max pooling后，在第二层卷积层，设定卷积核个数为20个，可以找出一些相对复杂的特征，如“横折”、“左半圆”、“右半圆”等特征，越往后，卷积核设定的数目越多，越能体现label的特征就越细致，就越容易分类出来，打个比方，如果你想分类出“0”的数字，你看到  这个特征，能推测是什么数字呢？只有越往后，检测识别的特征越多，试过能识别    这几个特征，那么我就能确定这个数字是“0”。


3.有stride_w和stride_h，后者表示的就是上下步长。如果用stride，则表示 stride_h=stride_w=stride。

手写数字识别的CNN网络结构

上面我们了解了卷积神经网络的基本结构后，现在来具体看一下在实际数据---手写数字识别中是如何操作的。上文中我定义了一个最基本的CNN网络。如下(代码详见[github](#))



```
1 def convolutional_neural_network_org(img):
2     # first conv layer
3     conv_pool_1 = paddle.networks.simple_img_conv_pool(
4         input=img,
5         filter_size=3,
6         num_filters=20,
7         num_channel=1,
8         pool_size=2,
9         pool_stride=2,
10        act=paddle.activation.Relu())
11    # second conv layer
12    conv_pool_2 = paddle.networks.simple_img_conv_pool(
13        input=conv_pool_1,
14        filter_size=5,
15        num_filters=50,
16        num_channel=20,
17        pool_size=2,
18        pool_stride=2,
19        act=paddle.activation.Relu())
20    # fully-connected layer
21    predict = paddle.layer.fc(
22        input=conv_pool_2, size=10, act=paddle.activation.Softmax())
23    return predict
```



那么它的网络结构是：

conv1-----> conv2----->fully Connected layer

非常简单的网络结构。第一层我们采取的是3*3的正方形卷积核，个数为20个，深度为1，stride为2，pooling尺寸为2*2，激活函数采取的为RELU；第二层只对卷积核的尺寸、个数和深度做了些变化，分别为5*5，50个和20；最后链接一层全连接，设定10个label作为输出，采用Softmax函数作为分类器，输出每个label的概率。

那么这个时候我考虑的问题是，既然上面我们已经了解了卷积核，改变卷积核的大小是否会对我的结果造成影响？增多卷积核的数目能够提高准确率？于是我做了个实验：

卷积核参数影响实验

	层数	卷积核尺寸	卷积核个数	通道数 (channel)	池化尺寸	步长	time	result
原始参数	第一层	5	20	1	2	2	4m13.729s	98.93%
	第二层	5	50	20	20	2		
第一次改进	第一层	3	20	1	2	2	4m20.276s	98.99%
	第二层	3	50	20	20	2		
第二次改进	第一层	3	20	1	2	2	4m48.579s	99.01%
	第二层	3	100	20	20	2		

- 第一次改进：仅改变第一层与第二层的卷积核数目的大小，其他保持不变。可以看到结果提升了0.06%

- 第二次改进：保持 3×3 的卷积核大小，仅改变第二层的卷积核数目，其他保持不变，可以看到结果相较于原始参数提升了0.08%

由以上结果可以看出，改变卷积核的大小与卷积核的数目会对结果产生一定影响，在目前手写数字识别的项目中，缩小卷积核尺寸，增加卷积核数目都会提高准确率。不过以上实验只是一个小测试，有兴趣的同学可以多做几次实验，看看参数带来的具体影响，下篇文章我们会着重分析参数的影响。

这篇文章主要介绍了神经网络的预备知识，卷积神经网络的常见的层及基本的计算过程，[看完后希望大家明白以下几个知识点](#)：

- 为什么卷积神经网络更适合于图像分类？相比于传统的神经网络优势在哪里？
- 卷积层中的卷积过程是如何计算的？为什么卷积核是有效的？
- 卷积核的个数如何确定？应该选择多大的卷积核对于模型来说才是有效的？尺寸必须为正方形吗？如果是长方形因该怎么做？
- 步长的大小会对模型的效果产生什么样的影响？垂直方向和水平方向的步长是否得设定为相同的？
- 为什么要采用池化层，Max Pooling有什么好处？
- Zero Padding有什么作用？如果已知一个feature map的尺寸，如何确定zero padding的数目？

上面的问题，有些在文章中已经详细讲过，有些大家可以根据文章的内容多思考一下。最后[给大家留几个问题思考一下](#)：

- 为什么改变卷积核的大小能够提高结果的准确率？卷积核大小对于分类结果是如何影响的？
- 卷积核的参数是怎么求的？一开始随机定义一个，那么后来是如何训练才能使这个卷积核识别某些特定的特征呢？
- 1×1 的卷积核有意义吗？为什么有些网络层结构里会采用 1×1 的卷积核？

[下篇文章我们会着重讲解以下几点](#)：

- 卷积核的参数如何确定？随机初始化一个数值后，是如何训练得到一个能够识别某些特征的卷积核的？
- CNN是如何进行反向传播的？
- 如何调整CNN里的参数？
- 如何设计最适合的CNN网络结构？
- 能够不用调用框架的api，手写一个CNN，并和paddlepaddle里的实现过程做对比，看看有哪些可以改进的？

ps：本篇文章是基于个人对CNN的理解来写的，本人能力有限，有些地方可能写的不是很严谨，如有错误或疏漏之处，请留言给我，我一定会仔细核实并修改的^_^！不接受无脑喷哦~此外，文中的图表结构均为自己所做，希望不要被人随意抄袭，可以进行非商业性质的转载，需要转载留言或发邮件即可，希望能够尊重劳动成果，谢谢！有不懂的也请留

言给我，我会尽力解答的哈~

作者：Charlotte77

出处：<http://www.cnblogs.com/charlotte77/>

本文以学习、研究和分享为主，如需转载，请联系本人，标明作者和出处，非商业用途！

关注【Charlotte数据挖掘】回复 '资料' 获取深度学习优质资料

分类：[深度学习](#)

标签：[深度学习](#)

好文要顶

关注我

收藏该文

[Charlotte77](#)
[关注 - 8](#)
[粉丝 - 3743](#)

96

2

推荐博客
[+加关注](#)

« 上一篇：[三个月教你从零入门深度学习](#)
» 下一篇：[【深度学习系列】数据预处理](#)

posted @ 2017-11-01 09:00 [Charlotte77](#) 阅读(178639) 评论(108) [编辑](#) [收藏](#)

[< Prev](#)

[1](#)

[2](#)

[3](#)

评论

#101楼 [楼主] 2019-12-24 11:22 | [Charlotte77](#)

@ 人生苦短我不会python
可以的

支持(0) 反对(0)

#102楼 2019-12-25 09:11 | 韩梦梵

哇 看起来高深莫测的 膜拜一下

支持(0) 反对(0)

#103楼 2020-03-19 22:25 | 爱吃面包的薯条

卷积那里
原文说第二行的数值绝对值最大，说明原始图片上对应的地方有一条水平方向的特征，
这个是绝对值之和的意思吗 这样的话为什么不是第一列呢 小姐姐看到能不能回答我一下 谢谢啦

支持(0) 反对(0)

#104楼 2020-04-16 10:18 | gii_walter

通过第一个卷积核计算后的feature_map是一个三维数据，在第三列的绝对值最大，说明原始图片上对应的地方有一条垂直方向的特征，即像素数值变化较大；而通过第二个卷积核计算后，第三列的数值为0，第二行的数值绝对值最大，说明原始图片上对应的地方有一条水平方向的特征。
-->小姐姐，为什么根据数值就能判断有特征？这里面有什么逻辑吗？

[支持\(0\)](#) [反对\(0\)](#)

#105楼 2020-04-16 10:33 | gii_walter

那么我们对其进行Max Pooling操作后，提取出的是真正能够识别特征的数值
--> 小姐姐，我比较菜哈。。。为什么说Max Pooling提取的是真正能够识别特征的数值？这是有什么依据吗？有没有min pooling呢？

[支持\(0\)](#) [反对\(0\)](#)

#106楼 2020-06-14 21:02 | douzujun

您好博主，为啥我感觉一开始最上面那个，参数个数是这样算的： $784 + (15784 + 151) + (1015 + 101)$ ，您 7841510 ，乘以10没有看懂是为什么， $w^{[l]}$ 的维数不是 $(n^{[l]}, n^{[l-1]})$ 吗，即，隐藏层 $w^{[1]} = (15, 784)$ ， $b^{[1]} = (15, 1)$ ，输出层 $w^{[2]} = (10, 15)$ ， $b^{[2]} = (10, 1)$ ，感谢回复

[支持\(0\)](#) [反对\(0\)](#)[< Prev](#) [1](#) [2](#) [3](#)[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)， [访问](#) 网站首页。

【推荐】超50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库

【推荐】了不起的开发者，挡不住的华为，园子里的品牌专区

【推荐】九大训练营同开 2020阿里云大数据独门绝学

【推荐】12知识点+20干货案例+110面试题，助你拿offer | Python面试宝典

相关博文：

- 深度学习之卷积神经网络(CNN)详解
- 卷积神经网络CNN总结
- 【深度学习系列】用PaddlePaddle和Tensorflow进行图像分类
- 【深度学习系列】卷积神经网络详解(二)——自己手写一个卷积神经网络
- 深度学习（一）——CNN算法流程
- » 更多推荐...

最新 IT 新闻：

- Twitter数据泄露案存分歧 欧盟或推迟对科技巨头隐私调查
- 阿里比腾讯多赚140亿 但人家会“炒股”啊
- @你的云南朋友 《采蘑菇模拟器》登陆Steam：仅售16元
- 微信小商店支持个人开店，推出小商店开放组件和多项新能力
- Google Maps增加了追踪加州山火的功能
- » 更多新闻...

Copyright © 2020 Charlotte77
Powered by .NET Core on Kubernetes