**1. 实验内容**

本实验首先对卷积神经网络的原理做介绍并配合一系列小的实验加深理解，之后通过简单的卷积神经网络实验实现手写数字识别。

**2. 实验目标**

通过本实验掌握卷积神经网络的原理，理解卷积神经网络在图像识别领域的成功应用，并为学习和理解更复杂的神经网络结构奠定基础。

**3. 实验知识点**

* 卷积神经网络

**4. 实验环境**

* python 3.6.5
* keras

**5. 预备知识**

* 传统机器学习知识
* Linux命令基本操作

[实验原理](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=110&guideProcID=918&doUp=true#collapseOne917)

计算机视觉的发展衍生出了新的神经网络结构和算法。

* 如果使用传统的全连接神经网络，处理较大图片时，过大的参数量使我们难以获取足够的数据来防止过拟合。
* 为了处理更复杂的图片，我们需要进行卷积运算。
* 卷积运算时卷积神经网络中非常重要的一块，我们将在后续的实验中，逐步学习。

### [实验步骤概览](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=110&guideProcID=918&doUp=true#collapseOne919)

从本步骤开始，将开始创建我们自己的卷积神经网络，按照步骤具体分为：

* 构建卷积层
* 构建池化层
* 构建全连接层
* 构建简单卷积网络

最后是一个完整的卷积神经网络的实例，利用卷积神经网络来进行手写识别。

现在，我们来开始学习如何构建卷积神经网络吧。

### [【实验步骤】CNN-边缘检测](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=110&guideProcID=918&doUp=true#collapseOne920)

卷积计算是卷积神经网络的最基本模块。

我们使用边缘检测作为入门的例子。

当我们处理一个图片时，我们可以对其进行垂直边缘检测，水平边缘检测等。

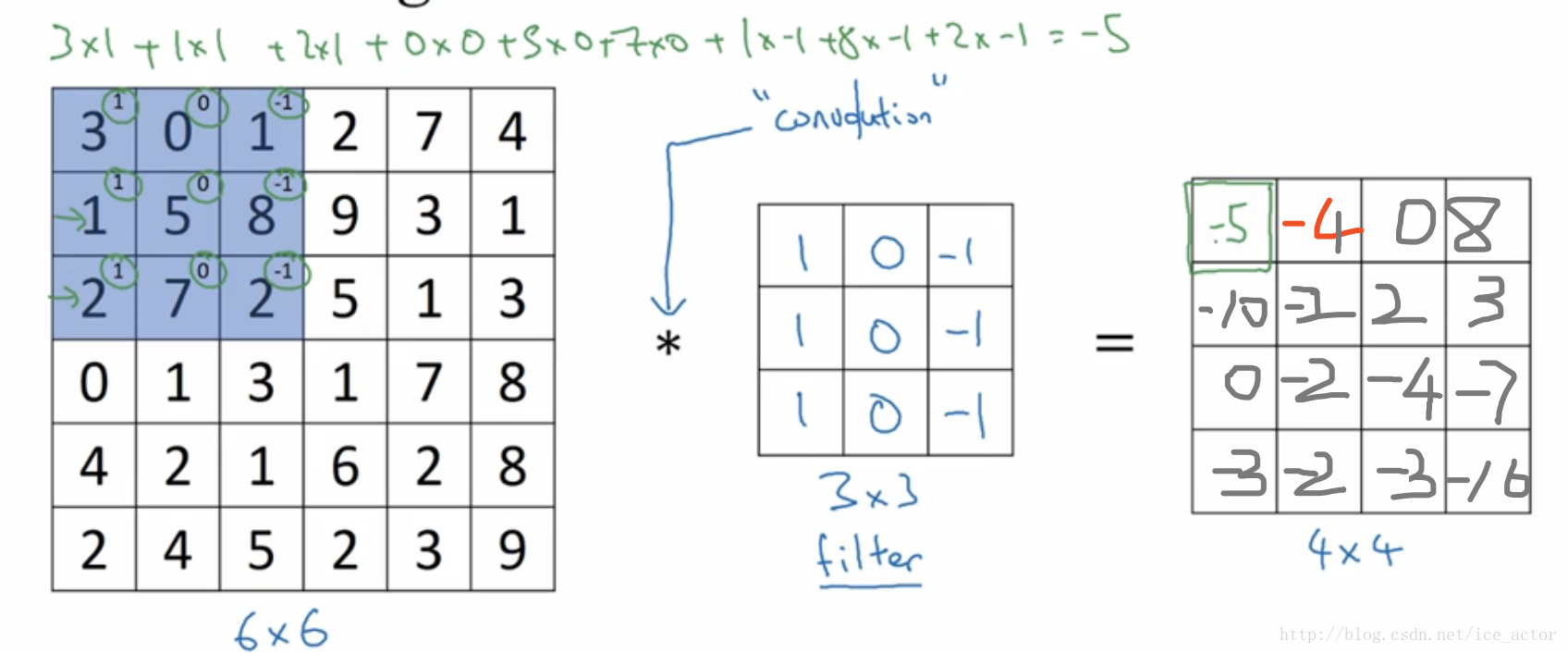
那么，如何对图片进行边缘检测呢？

我们先来从理论上理解一个卷积计算的过程：

下图左边是一个6\*6\*1（height\*width\*channel）的灰度图像，我们构造一个3\*3的矩阵，称做过滤器filter。

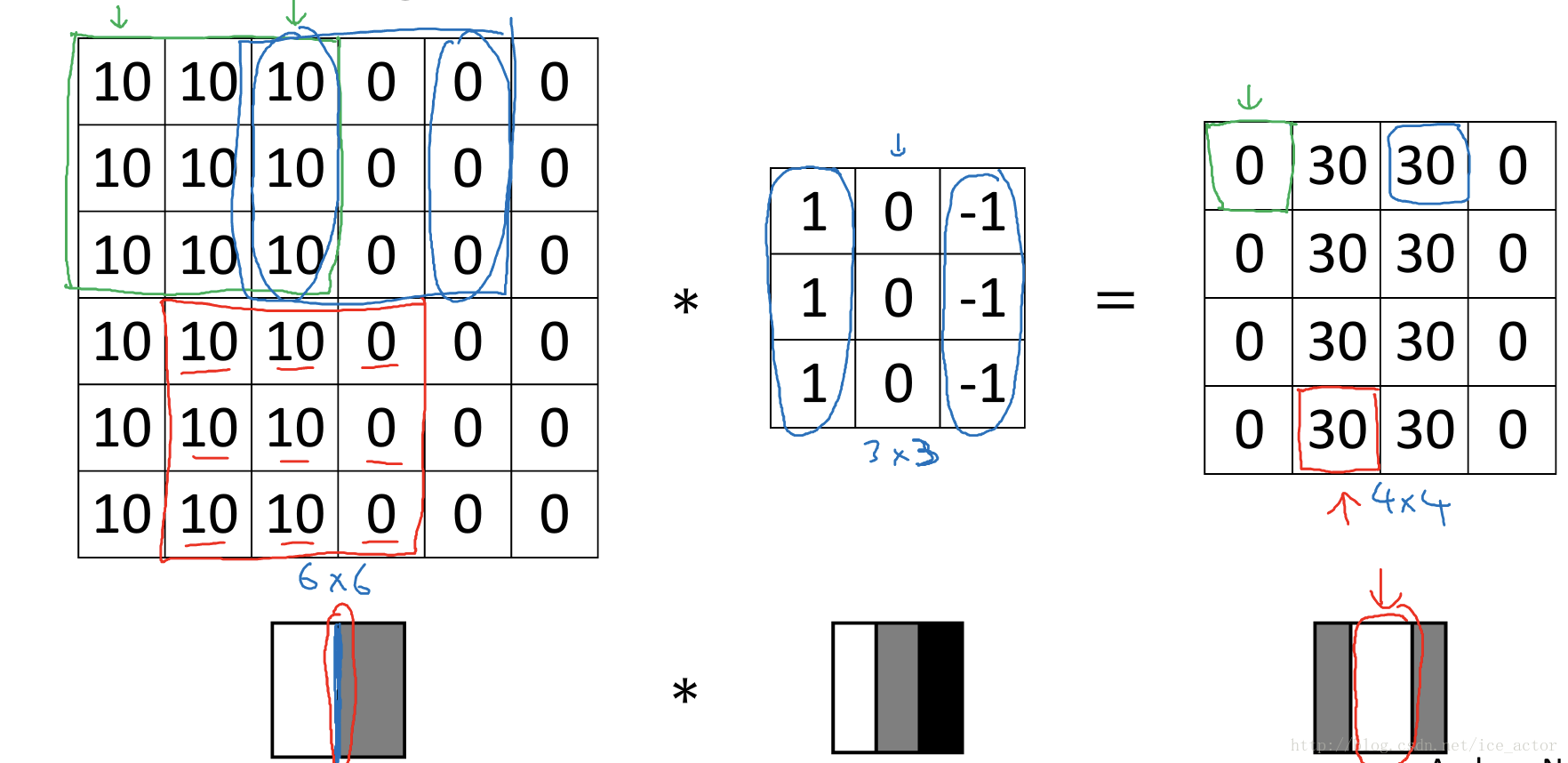
对该图像进行卷积运算，如图所示，输出的结果是4\*4的矩阵，那么是如何得到这个结果的？

对于左上角的第一个位置，我们对图片使用3\*3的过滤器，将其覆盖在输入图像上，如图中蓝色区域所示。对其进行元素乘法运算。每次计算后，将蓝色区域向右移动一个步长（后面讲），计算下一个模块。



那么，卷积计算是如何进行边缘检测的呢？

在下图中，0表示暗色区域，10表示亮色区域，我们可以很明确的看出亮色到暗色的过滤区域，同样用3\*3的过滤器来对图像进行处理，我们可以看到，结果很好的反应出了过渡边缘。

可能会觉得这个边缘有点粗，是由于当前图片过小，对于较大图片会有更好的效果。  


同样的对过滤器进行转置，就可以得到一个水平边缘检测器。

对应不同的深度学习框架，卷积计算会有不同的实现方法。

|  |  |
| --- | --- |
| 框架 | 函数方法 |
| pyton | conv-forward |
| tensorflow | tf.nn.conv2d |
| keras | conv2D |

在本次实验中，我们使用keras框架来实现卷积计算。

### [【实验步骤】CNN-padding](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=110&guideProcID=918&doUp=true#collapseOne921)

通过上一节的实验，我们观察到：

    1.  每进行一次卷积计算，图像的大小会缩小一次。如果有100层的卷积层，每层都进行卷积计算，那么输出图像会一直在缩小。

    2. 在进行卷积计算的过程中，有些像素多次参与计算，但边缘像素大部分信息丢失。

为了解决这两个问题，我们引入了padding，即填充。

对输入图像进行填充，使得边缘像素能够更多的参与运算。

常见的padding有两种模式：

 1. valid：no padding。不填充

 2. same：对输入图像进行填充，使得输出图像的大小与原输入图像大小相同。

我们可以考虑一下，在padding的same模式下，如何设置padding个数来满足该要求。

假设输入图片大小为n\*n，过滤器为f\*f，padding为x，此处默认步长为1。

输出图像的大小为：(n+2x-f+1)\*(n+2x-f+1)

令n+2x-f+1=n，可以得出x=(f-1)/2。

我们只需要对上节的实验进行参数的修改即可实现该功能。

### [【实验步骤】CNN-卷积步长](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=110&guideProcID=918&doUp=true#collapseOne922)

卷积的步长是构建卷积神经网络的另一个基本操作。

在上两节的实验中，我们默认将步长(stride)设为1，即每次向右或向下移动一步。

在本节实验中，我们将步长修改为2。即每次向右或者向下移动2步。

对于7\*7的输入图像，我们选择3\*3的过滤器，步长设为s=2,padding为p。

那么输出图像的大小应为[(n+2p-f)/s]+1  \*  [(n+2p-f)/s]+1，即（7+0-3）/2 +1 =3

当计算结果不是整数时，我们应对结果进行向下取整，确保计算范围中都为有效值。

### [CNN-构建卷积层](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=110&guideProcID=918&doUp=true#collapseOne923)

到此为止，我们已经了解一些基础的知识。

那么，如何构建卷积神经网络的卷积层呢？

在实际生活中，我们在处理图像时不仅仅是检测一种类型的特征，可能同时需要检测垂直边缘，水平边缘，45°边缘等等。

这时，我们可以选择c个过滤器，分别与输入图像进行卷积计算，得到输出图像n\*n\*c。

我们可以下面的图中了解到卷积层的完整计算过程。

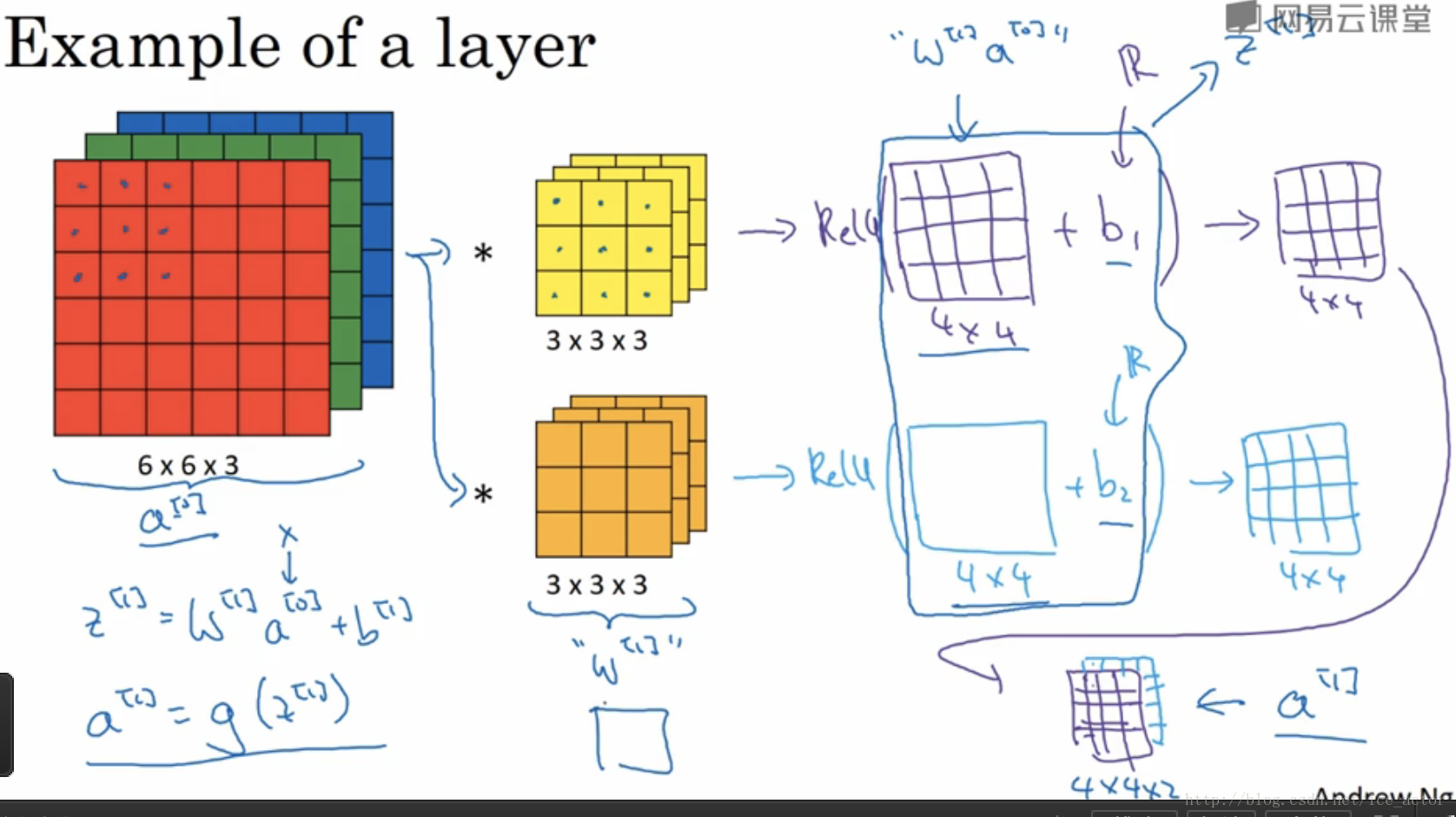
输入图像通过两个过滤器分别得到了两个矩阵。对于两个矩阵加上不同的偏差b1和b2，并对和进行非线性的Relu变换，即激活函数此处选用了relu，得到了一个新的矩阵。即该卷积层的输出图像。

公式表示如下：

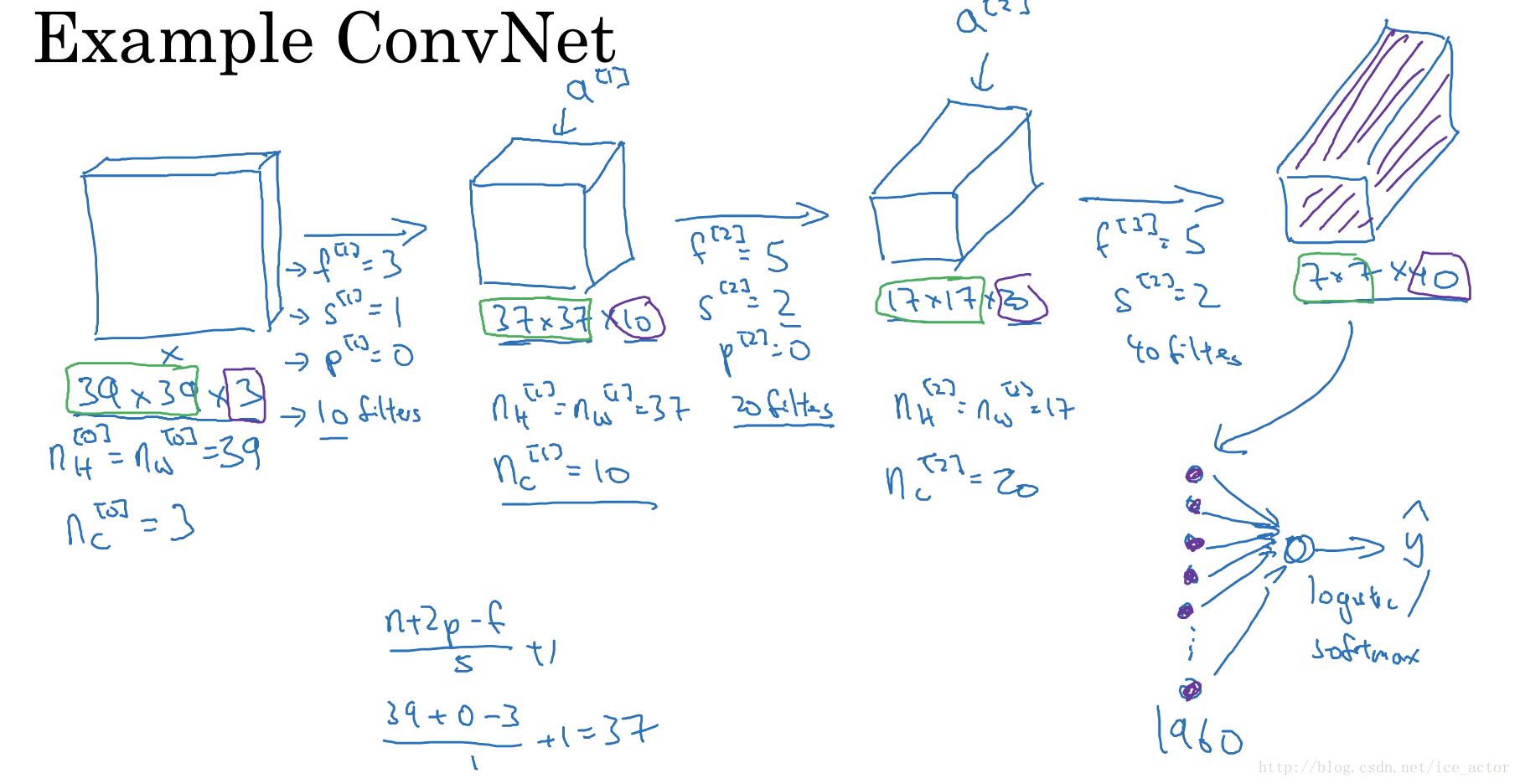
z[1]=w[1]a[0]+b[1]

a[1]=g(z[1])

其中输入图像为a[0],过滤器用w[1]表示，对图像进行线性变化并加入偏差得到矩阵z[1]，a[1]是应用Relu激活后的结果。



我们需要注意，输出图像的通道数即为上一层卷积层的过滤器个数n。



from keras.models import Sequential  
from keras.layers.convolutional import Conv2D  
  
model = Sequential()  
mode1.add(Conv2D(30, (5, 5), input\_shape=(28, 28,1), activation='relu',padding='valid'))

### [CNN-构建池化层](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=110&guideProcID=918&doUp=true#collapseOne924)

在卷积神经网络中，除了卷积层，也可以使用池化层来缩减模型大小，提高计算速度。

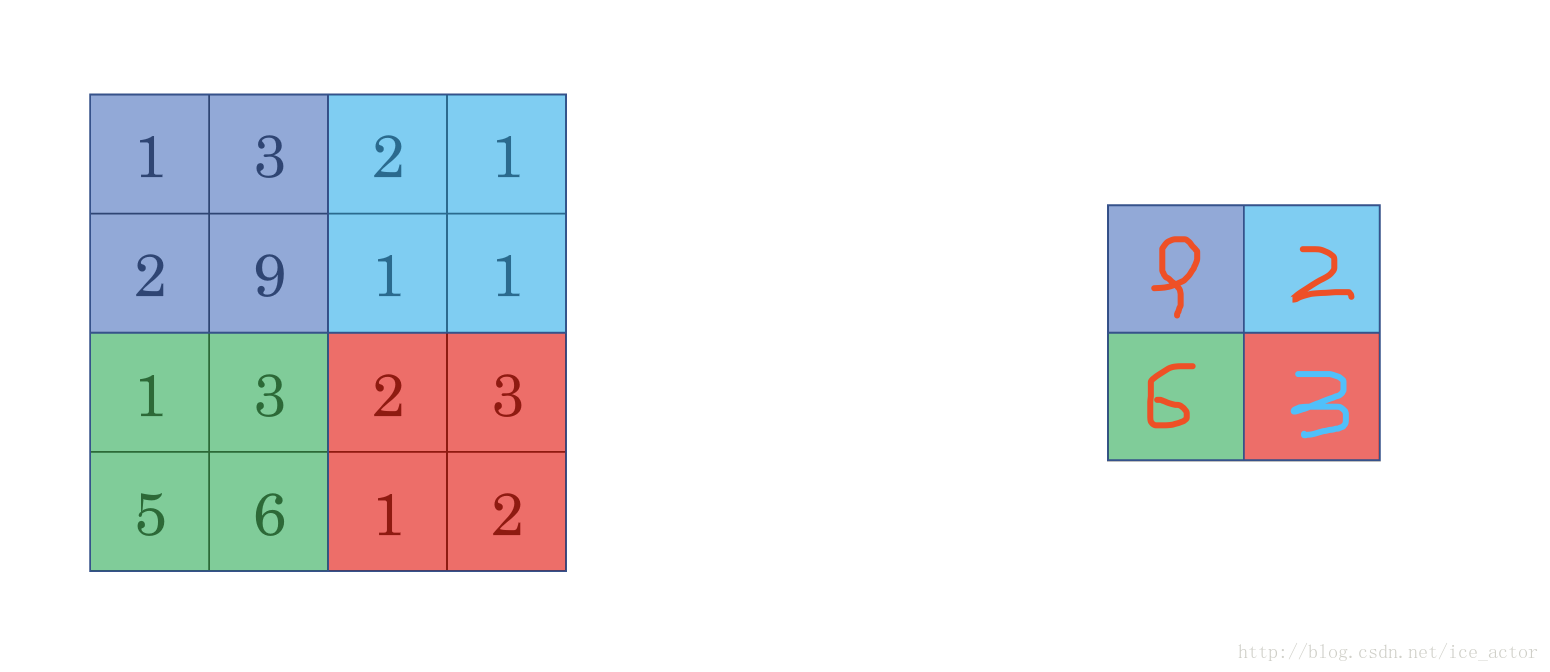
同时可以提高所提取特征的鲁棒性。

那么，何为池化？

常见的池化类型包括：最大池化（maxpooling）和平均值池化（arrpooling）

我们先来看最大池化的例子。

以下图为例，把4\*4的图像分割成4个不同的区域，然后输出每个区域的最大值，这就是最大池化所做的事情。其实这里我们选择了2\*2的过滤器，步长为2。在一幅真正的图像中提取最大值可能意味着提取了某些特定特征，比如垂直边缘、一只眼睛等等。



平均池化和最大池化唯一的不同是，它计算的是区域内的平均值而最大池化计算的是最大值。在日常应用使用最多的还是最大池化。

池化层在卷积神经网络中，无需学习，需要静态属性。

from keras.models import Sequential  
from keras.layers.convolutional import MaxPooling2D  
  
model = Sequential()  
model.add(MaxPooling2D(pool\_size=(2, 2)))

### [CNN-构建全连接层](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=110&guideProcID=918&doUp=true#collapseOne925)

全连接层，如其名。

将上一层的输出图像平铺为向量，进行全连接处理。

处理后的输出数据输出到logistic或softmax来得到相应的预测值

from keras.models import Sequential  
from keras.layers import Dense  
from keras.layers import Flatten  
  
  
model = Sequential()  
model.add(Flatten())  
model.add(Dense(128, activation='relu'))

### [CNN-构建简单卷积网络](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=110&guideProcID=918&doUp=true#collapseOne926)

1. 输入图像为32\*32\*3的rgb图像，我们也可以使用mnist的数据集，即28\*28\*1.

2. 假设第一层使用的过滤器为5\*5，步长为1，padding为0，过滤器个数为6，输出图像大小为28\*28\*6。

3. 对其加上偏差，并应用relu非线性函数，得到第一层卷积层的输出结果，标记为CONV1。

4. 然后构建一个池化层，此处选择最大池化，大小为2\*2，步长为2.意味着输出图像的高度和宽度会减少一半，由于信道数量保持不变，即为14\*14\*6。该输出标记为POOL1。

5. 我们将卷积层和池化层共同作为一个卷积层，记为Layer1.

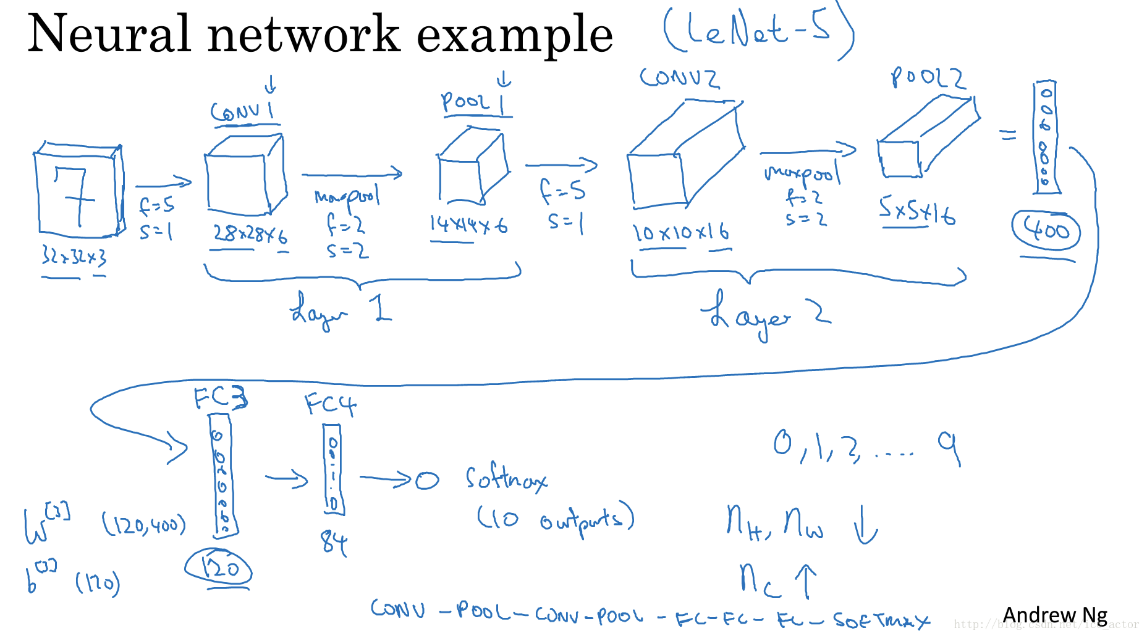
6. 再对该模型应用第二种过滤器，大小为10\*10，个数为16个。最大池化大小为2\*2，步长为2.得到第二层的输出结果为5\*5\*16，标记为POOL2.

7. 将POOL2平整化为一个大小为400的一维向量。并利用这400个单元构建下一层

8. 假设下一层含有120个单元，即我们的第一个全连接层，记为F3。标准的神经网络层。最后输出120个维度。

9. 再对这120个单元添加一个全连接层，记为F4。

10. 最后将F4的输出单元填充softmax单元，得到预测值。



from keras.models import Sequential  
from keras.layers import Dense  
from keras.layers import Dropout  
from keras.layers import Flatten  
from keras.layers.convolutional import Conv2D  
from keras.layers.convolutional import MaxPooling2D  
  
def build\_cnn:  
      
    model = Sequential()  
      
    model.add(Conv2D(6, (5,5), input\_shape=(32, 32,3), activation='relu',padding='valid'))  
    model.add(MaxPooling2D(pool\_size=(2, 2)))  
  
    model.add(Conv2D(16, (5, 5), activation='relu'))  
    model.add(MaxPooling2D(pool\_size=(2, 2)))  
      
    model.add(Flatten())  
    model.add(Dense(120, activation='relu'))  
    model.add(Flatten())      
    model.add(Dense(84, activation='relu'))      
    model.add(Dense(num\_classes, activation='softmax'))  
    # Compile model  
    model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])  
    return model

### [CNN-卷积神经网络用于手写识别](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=110&guideProcID=918&doUp=true#collapseOne927)

在本节实验中，我们利用keras库来构建卷积神经网络用于手写识别。

我们的卷积神经网络结构为：

layer1：conv2D，maxpool  
dropout  
layer2：conv2D，maxpool  
dropout  
layer3：全连接层  
softmax

实现代码：

设置运行环境

import os  
os.environ['TF\_CPP\_MIN\_LOG\_LEVEL']='2'

导入实验所需要的包

import numpy  
from keras.datasets import mnist  
from keras.models import Sequential  
from keras.layers import Dense  
from keras.layers import Dropout  
from keras.layers import Flatten  
from keras.layers.convolutional import Conv2D  
from keras.layers.convolutional import MaxPooling2D  
from keras.utils import np\_utils  
import matplotlib.pyplot as plt  
from keras.constraints import maxnorm  
from keras.optimizers import SGD

seed( ) 用于指定随机数生成时所用算法开始的整数值，如果使用相同的seed( )值，则每次生成的随即数都相同，如果不设置这个值，则系统根据时间来自己选择这个值，此时每次生成的随机数因时间差异而不同。

设置可再生随机数

seed = 7  
numpy.random.seed(seed)

加载mnist数据

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

为了使用keras库，将图像大小调整为[samples][width][height][channel]

X\_train = X\_train.reshape(X\_train.shape[0],28, 28,1).astype('float32')  
X\_test = X\_test.reshape(X\_test.shape[0], 28, 28,1).astype('float32')

数据正则化，避免过拟合

X\_train = X\_train / 255  
X\_test = X\_test / 255

对输出热编码

y\_train = np\_utils.to\_categorical(y\_train)  
y\_test = np\_utils.to\_categorical(y\_test)  
num\_classes = y\_test.shape[1]

定义卷积神经网络，创建模型

def larger\_model():  
      
    # create model  
    model = Sequential()

在模型中添加卷积层，池化层和全连接层。

keras提供了接口函数可以很方便的实现，此处使用了dropout层，可按实际训练情况修改。

    model.add(Conv2D(30, (5, 5), input\_shape=(28, 28,1), activation='relu',padding='valid'))  
    model.add(MaxPooling2D(pool\_size=(2, 2)))  
    model.add(Dropout(0.4))  
    model.add(Conv2D(15, (3, 3), activation='relu'))  
    model.add(MaxPooling2D(pool\_size=(2, 2)))  
    model.add(Dropout(0.4))  
    model.add(Flatten())  
    model.add(Dense(128, activation='relu'))  
    model.add(Dropout(0.4))  
    model.add(Dense(50, activation='relu'))  
    model.add(Dropout(0.4))  
    model.add(Dense(num\_classes, activation='softmax'))

编译模型

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])  
return model

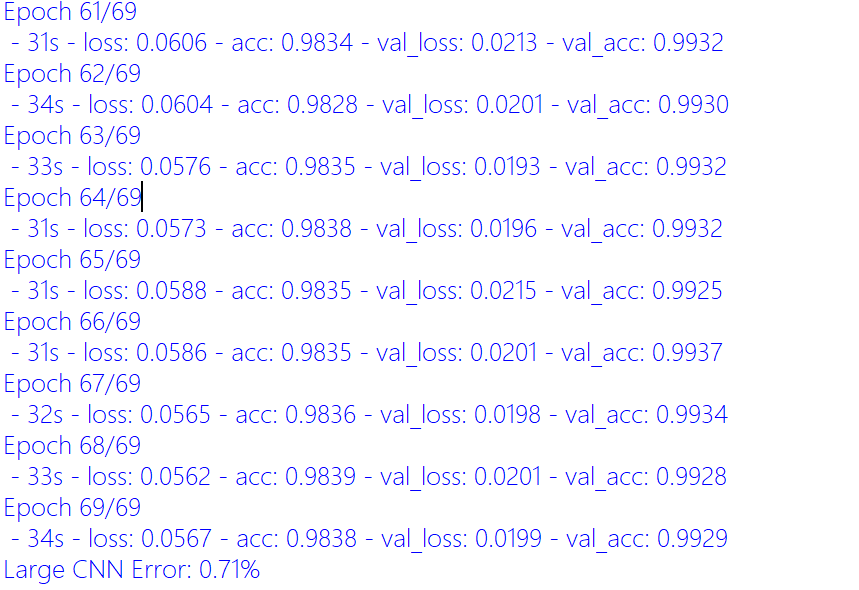
训练模型

* fit函数即为训练模型函数。参数中validation\_data指定的验证集，epochs指训练次数，batch\_size指每次梯度下降时每个batch包含的样本数。verbose代表日志显示：0为不在标准输出流输出日志信息，1为输出进度条记录，2为每个epoch输出一行记录。
* evaluate函数即为按batch计算在某些输入数据上模型的误差。参数类型与fit相同。

使用"categorical\_crossentropy"作为目标函数时,标签应该为多类模式,即one-hot编码的向量,

# Fit the model  
model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=69, batch\_size=200, verbose=2)  
# Final evaluation of the model  
scores = model.evaluate(X\_test, y\_test, verbose=0)  
print("Large CNN Error: %.2f%%" % (100-scores[1]\*100))

训练结果为：



迭代的结果中，loss和acc为训练集的结果，val\_loss和val\_acc为验证机的结果。可以看到我们的模型训练后，正确率为99.29%

### [实验总结](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=110&guideProcID=918&doUp=true#collapseOne928)

通过本实验，加深对卷积神经网络的理解。

掌握使用keras框架创建简单的卷积神经网络。

同时，我们需要知道，为什么使用卷积神经网络。

相比于传统的神经网络，卷积神经网络使用较少的参数并减少连接项。

卷积神经网络通过参数共享（即过滤器）和稀疏连接（卷积操作）来减少参数，提高计算速度

### [参考文献及延伸阅读](http://demo.cjudge.net:8081/admin/courseAdmin/expAdmin/guides/editExpGuide.jsp?guideID=110&guideProcID=918&doUp=true#collapseOne929)

**参考资料：**

1. 郑泽宇，梁博文，顾思宇.TensorFlow：实战Google深度学习框架[M]. 电子工业出版社, 2018.

**延伸阅读：**

1. Ian Goodfellow,Yoshua Bengio,Aaron Courville. 深度学习[M]..人民邮电出版社，2017.