1. 卷积神经网络

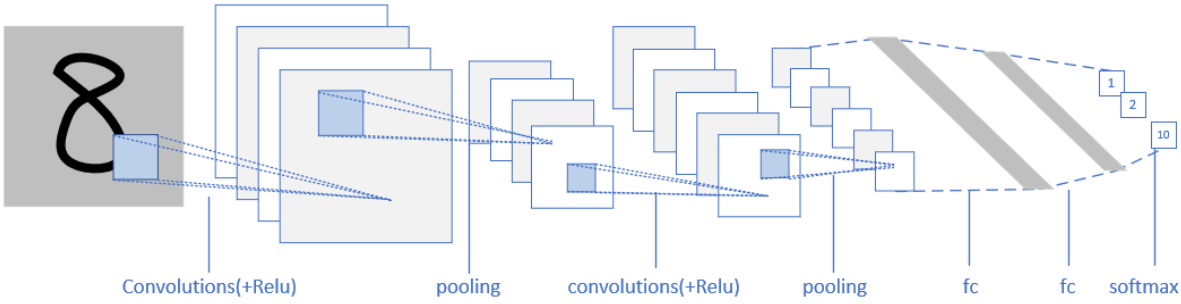
1.1 概述

1.1.1 卷积神经网络的能力

卷积神经网络（CNN，Convolutional Neural Net)是神经网络的类型之一，在图像识别和分类领域中取得了非常好的效果，比如识别人脸、物体、交通标识等，这就为机器人、自动驾驶等应用提供了坚实的技术基础。

1.1.2 卷积神经网络的典型结构

一个典型的卷积神经网络的结构如图所示：



这里是典型的数字识别,分析其中的层次结构如下：

1. 原始的输入是一张图片，可以是彩色的，也可以是灰度的或黑白的。这里假设是只有一个通道的图片，目的是识别0~9的手写体数字；
2. 第一层卷积，我们使用了4个卷积核，得到了4张feature map；激活函数层没有单独画出来，这里我们紧接着卷积操作使用了Relu激活函数；
3. 第二层是池化，使用了Max Pooling方式，把图片的高宽各缩小一倍，但仍然是4个feature map；
4. 第三层卷积，我们使用了4x6个卷积核，其中4对应着输入通道，6对应着输出通道，从而得到了6张feature map，当然也使用了Relu激活函数；
5. 第四层再次做一次池化，现在得到的图片尺寸只是原始尺寸的四分之一左右；
6. 第五层把第四层的6个图片展平成一维，成为一个fully connected层；
7. 第六层再接一个小一些的fully connected层；
8. 最后接一个softmax函数，判别10个分类。

所以，在一个典型的卷积神经网络中，会至少包含以下几个层：

- 卷积层
- 激活函数层
- 池化层
- 全连接分类层

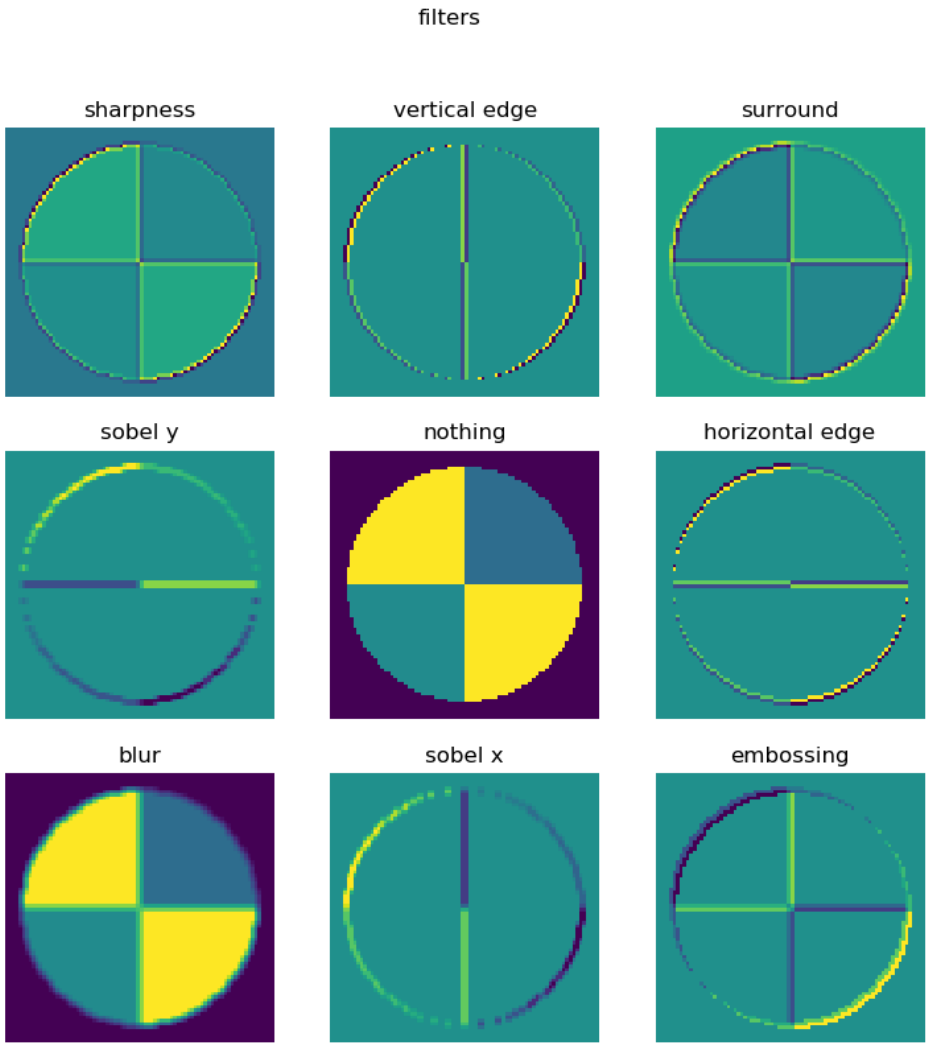
1.2.3 卷积核

定义： 就是一个小矩阵

在卷积层中，我们会用输入数据与卷积核相乘，得到输出数据，就类似全连接层中的Weights一样，所以卷积核里的数值，也是通过反向传播的方法学习到的。

卷积核的具体作用：

这是一个例子：



如图所示的内容，是使用9个不同的卷积核在同一张图上运算后得到的结果，下表是具体的卷积核的数据

	1	**2**	**3**
1	0,-1, 0 -1, 5,-1 0,-1, 0	0, 0, 0 -1, 2,-1 0, 0, 0	1, 1, 1 1,-9, 1 1, 1, 1
	sharpness	vertical edge	surround
2	-1,-2, -1 0, 0, 0 1, 2, 1	0, 0, 0 0, 1, 0 0, 0, 0	0,-1, 0 0, 2, 0 0,-1, 0
	sobel y	nothing	horizontal edge
3	0.11,0.11,0.11 0.11,0.11,0.11 0.11,0.11,0.11	-1, 0, 1 -2, 0, 2 -1, 0, 1	2, 0, 0 0,-1, 0 0, 0,-1
	blur	sobel x	embossing

其中第五张是原图，下面是 各个卷积核的作用

序号	名称	说明
1	锐化	如果一个像素点比周围像素点亮，则此算子会令其更亮
2	检测竖边	检测出了十字线中的竖线，由于是左侧和右侧分别检查一次，所以得到两条颜色不一样的竖线
3	周边	把周边增强，把同色的区域变弱，形成大色块
4	Sobel-Y	纵向亮度差分可以检测出横边，与横边检测不同的是，它可以使得两条横线具有相同的颜色，具有分割线的效果
5	Identity	中心为1四周为0的过滤器，卷积后与原图相同
6	横边检测	检测出了十字线中的横线，由于是上侧和下侧分别检查一次，所以得到两条颜色不一样的横线
7	模糊	通过把周围的点做平均值计算而“杀富济贫”造成模糊效果
8	Sobel-X	横向亮度差分可以检测出竖边，与竖边检测不同的是，它可以使得两条竖线具有相同的颜色，具有分割线的效果
9	浮雕	形成大理石浮雕般的效果

1.2.4 卷积过后的运算

就像我们最开始说的，卷积神经网络通过反向传播而令卷积核自我学习，找到分布在图片中的不同的feature，最后形成的卷积核中的数据。但是如果想达到这种效果，只有卷积层的话是不够的，还需要激活函数、池化等操作的配合。

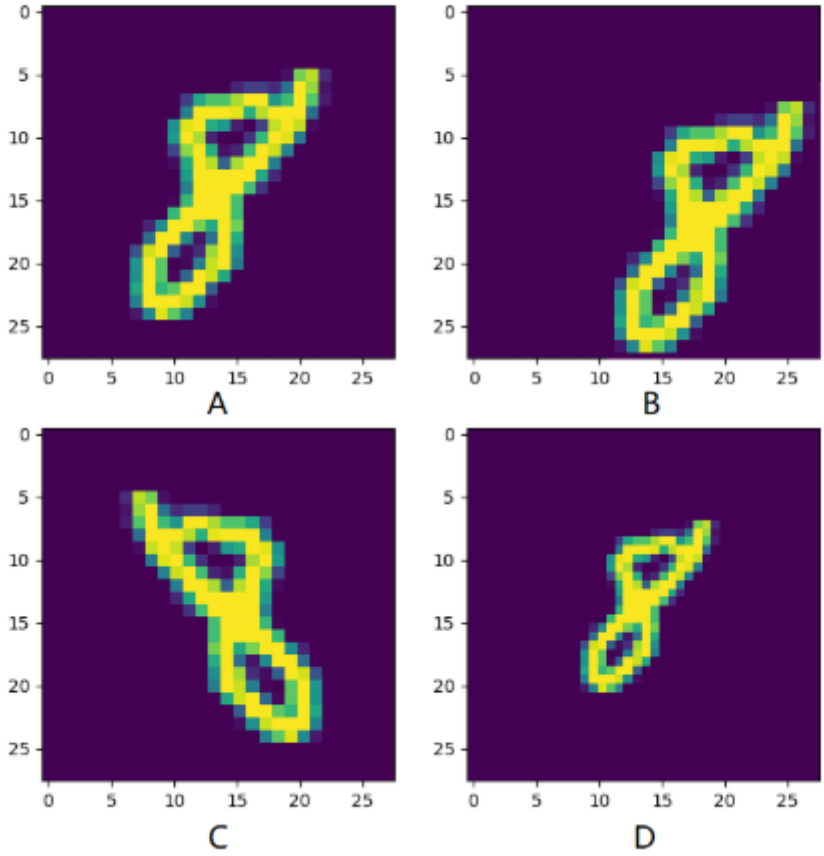
1.2.5 卷积神经网络的学习

在最后一层的池化后面，把所有特征数据变成一个一维的**全连接层**，然后就和普通的**深度全连接网络**一样了，通过在最后一层的softmax分类函数，以及多分类交叉熵函数，对比图片的OneHot编码标签，回传误差值，从全连接层传回到池化层，通过激活函数层再回传给卷积层，对卷积核的数值进行梯度更新，实现卷积核数值的自我学习。

在很多情况下，对于图片来说，平移，旋转，放大和缩小我们都是可以通过人脸一眼看出来的，但是对于卷积神经网络来说，它会怎么处理呢?

下面给出例子：

[8]



- 平移不变性

  对于原始图A，平移后得到图B，对于同一个卷积核来说，都会得到相同的特征，这就是卷积核的权值共享。但是特征处于不同的位置，由于距离差距较大，即使经过多层池化后，也不能处于近似的位置。此时，后续的全连接层会通过权重值的调整，把这两个相同的特征看作同一类的分类标准之一。如果是小距离的平移，通过池化层就可以处理了。

  使用： 池化层处理或者全连接层权重调整

- 旋转不变性

  对于原始图A，有小角度的旋转得到C，卷积层在A图上得到特征a，在C图上得到特征c，可以想象a与c的位置间的距离不是很远，在经过两层池化以后，基本可以重合。所以卷积网络对于小角度旋转是可以容忍的，但是对于较大的旋转，需要使用数据增强来增加训练样本。一个极端的例子是当6旋转90度时，谁也不能确定它到底是6还是9。

  使用： 多层池化处理，如果旋转角度过大，需要新数据进行训练

- 尺度不变性

  对于原始图A和缩小的图D，人们可以毫不犹豫的进行分辨；池化在这里是不是有帮助呢？没有！因为神经网络对A做池化的同时，也会用相同的方法对D做池化，这样池化的次数一致，最终D还是比A小。如果我们有多个卷积视野，相当于从两米远的地方看图A，从一米远的地方看图D，那么A和D就可以很相近似了。

  使用：Inception的想法，用不同尺寸的卷积核去同时寻找同一张图片上的特征。

1.2 卷积的向前计算

我们始终要记住，卷积核就是一个矩阵

数学定义：

连续定义

$$h(x) = (f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt \quad (1)$$

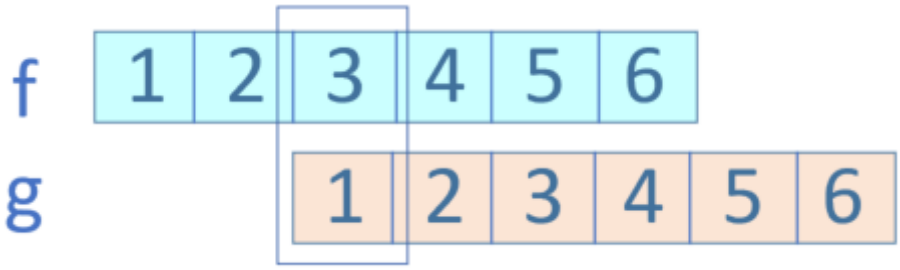
卷积与傅里叶变换有着密切的关系。利用这点性质，即两函数的傅里叶变换的乘积等于它们卷积后的傅里叶变换，能使傅里叶分析中许多问题的处理得到简化。

离散定义

$$h(x) = (f * g)(x) = \sum_{t=-\infty}^{\infty} f(t)g(x-t) \quad (2)$$

1.2.1 一维卷积的实例

这里有点像，两个数组，我们找两个数组内元素的和为一个定值的编程题，我们确定好一个数组，另外一个数组按照每步一位的进行平移，之后去找对应的确定是值就好，如下图：



1.2.2 单入单出的二维卷积

对于二维卷积，就是一个矩阵，一般用在二维图片上做卷积

如果把图像Image简写为I，把卷积核Kernal简写为K，则目标图片的第(i,j)个像素的卷积值为：

$$h(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \quad (3)$$

可以看出，这和一维情况下的公式2是一致的。从卷积的可交换性，我们可以把公式3等价地写作：

$$h(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \quad (4)$$

公式4的成立，是因为我们将Kernal进行了翻转。在神经网络中，一般会实现一个互相关函数(corresponding function)，而卷积运算几乎一样，但不反转Kernal：

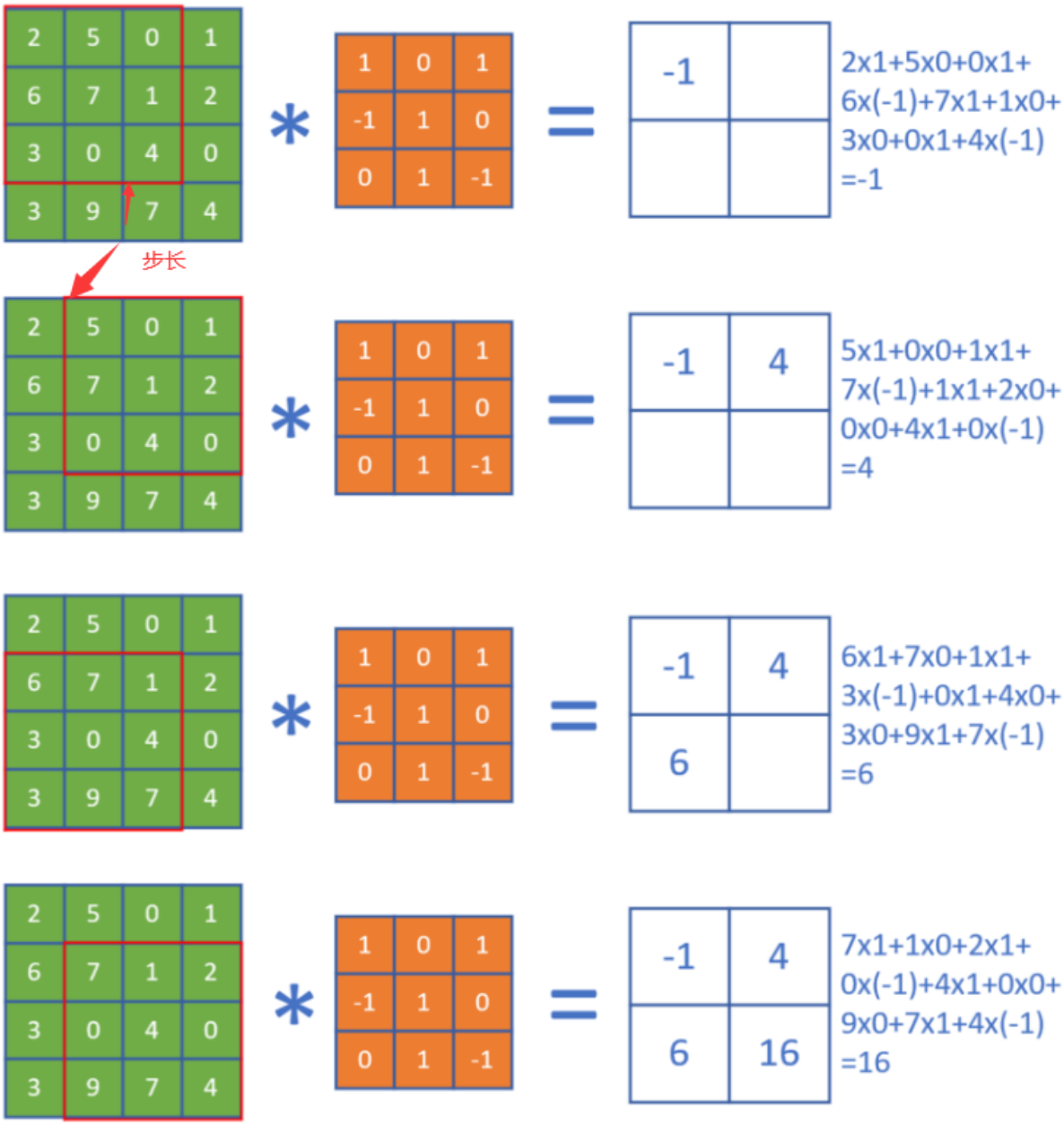
$$h(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \quad (5)$$

在图像处理中，自相关函数和互相关函数定义如下：

- 自相关：设原函数是f(t)，则$h = f(t) \star f(-t)$，其中$\star$表示卷积
- 互相关：设两个函数分别是f(t)和g(t)，则$h = f(t) \star g(-t)$

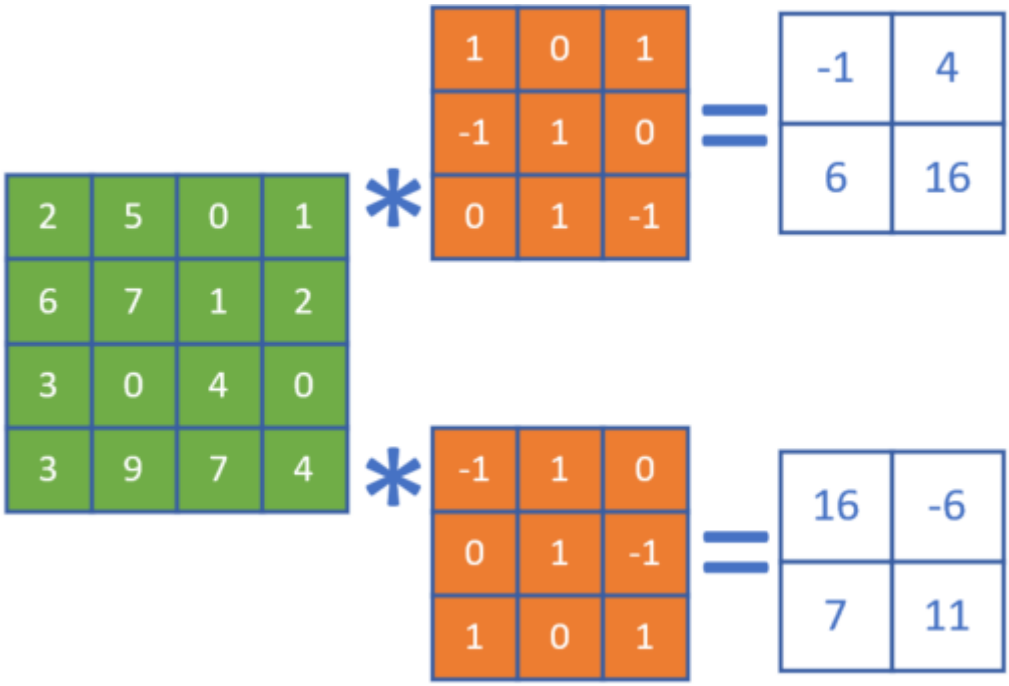
在传统图像处理上，我们可以认为卷积是利用某些设计好的参数组合（卷积核）去提取图像空域上相邻的信息。

卷积的计算和我们理解的矩阵相乘是不一样的，具体见下图：



1.2.3 单入多出的升维卷积

输入数据为一维，多个卷积核进行操作



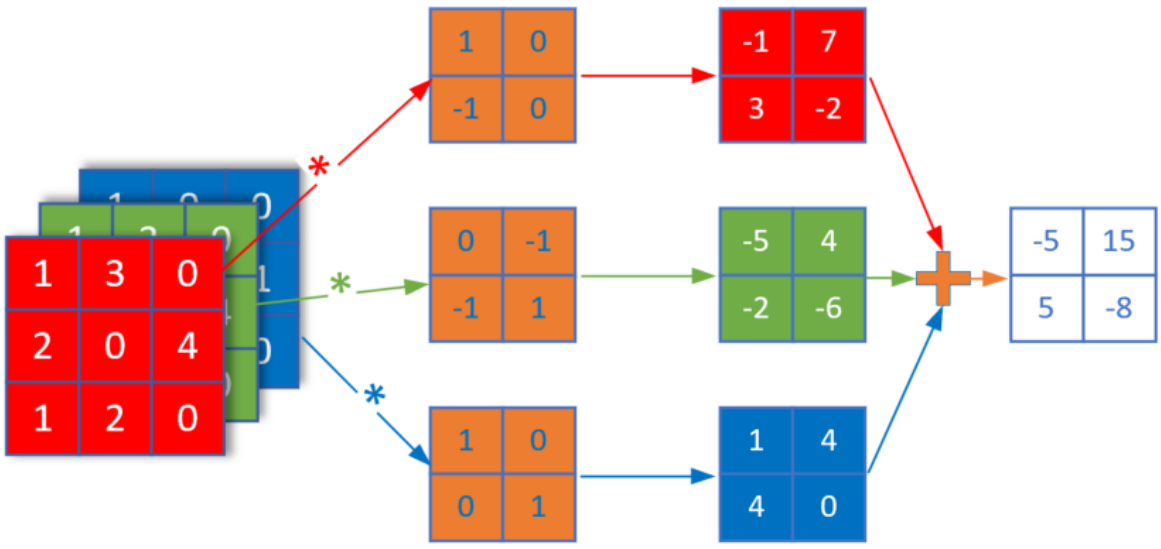
1.2.4 多入单出的降维卷积

一张图片，通常是彩色的，具有红绿蓝三个通道。可以有两个选择来处理：

1. 变成灰度的，每个像素只剩下一个值，就可以用二维卷积
2. 对于三个通道，每个通道都使用一个卷积核，分别处理红绿蓝三种颜色的信息

显然第2种方法可以从图中学习到更多的特征，于是出现了三维卷积，即有三个卷积核分别对应书的三个通道，三个子核的尺寸是一样的，比如都是2x2，这样的话，这三个卷积核就是一个3x2x2的立体核，称为过滤器Filter，所以称为三维卷积。

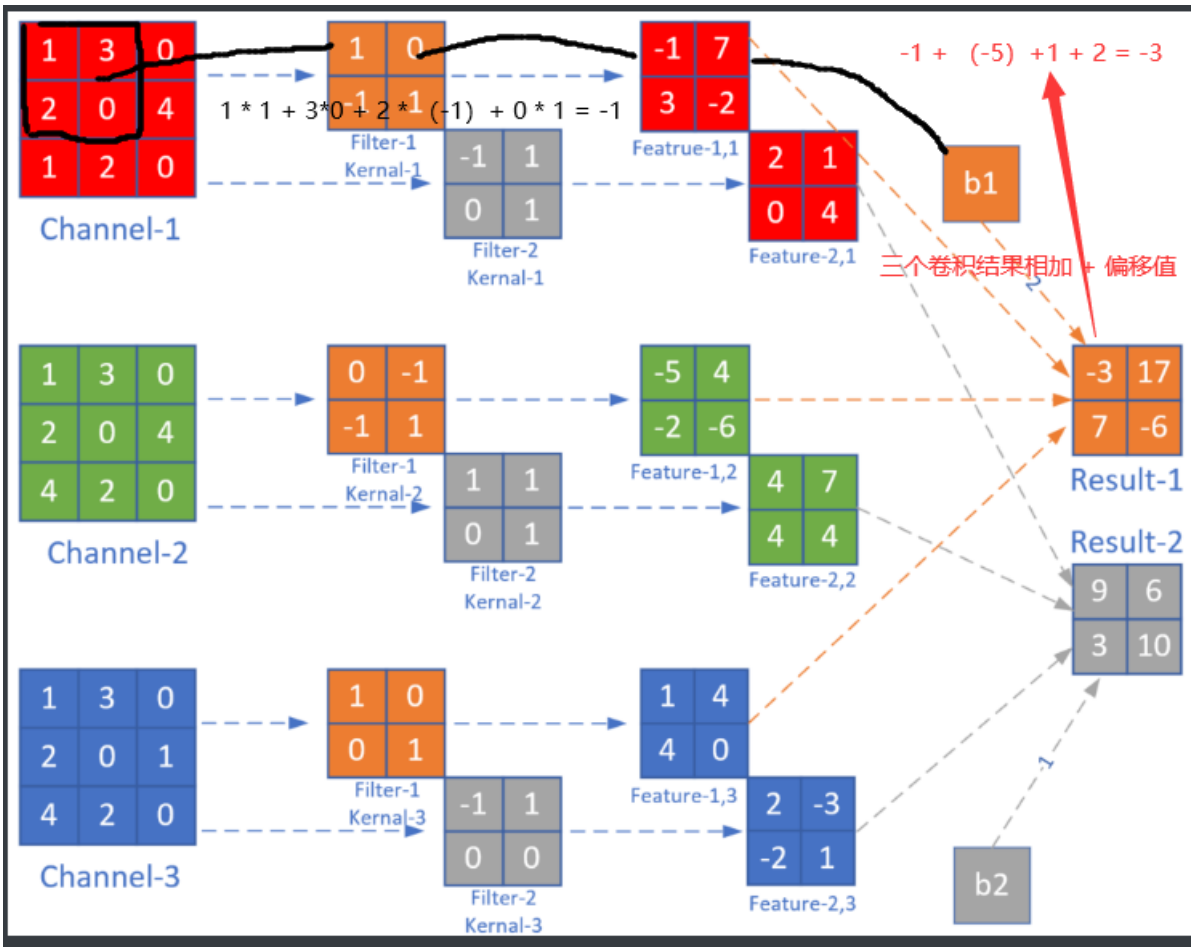
这里出现了新的名词：过滤器，在后面的学习里面我们可以看到，过滤器就是由卷积核组成的，过滤器可以组成一个过滤器组，卷积核也可以构成卷积核组，如下就是：**是一个过滤器Filter内含三个卷积核Kernal**



在上图中，每一个卷积核对应着左侧相同颜色的输入通道，三个过滤器的值并不一定相同。对三个通道各自做卷积后，得到右侧的三张特征图，然后再按照原始值不加权地相加在一起，得到最右侧的白色特征图，这张图里面已经把三种颜色的特征混在一起了，所以画成了白色，表示没有颜色特征了。

虽然输入图片是多个通道的，或者说是三维的，但是在相同数量的过滤器的计算后，相加在一起的结果是一个通道，即2维数据，所以称为降维。这当然简化了对多通道数据的计算难度，但同时也会损失多通道数据自带的颜色信息。

1.2.5 多入多出的同维卷积



第一个过滤器Filter-1为棕色所示，它有三卷积核(Kernal)，命名为Kernal-1，Keanrl-2，Kernal-3，分别在红绿蓝三个输入通道上进行卷积操作，生成三个2x2的输出Feature-1,n。然后三个Feature-1,n相加，并再加上b1偏移值，形成最后的棕色输出Result-1。

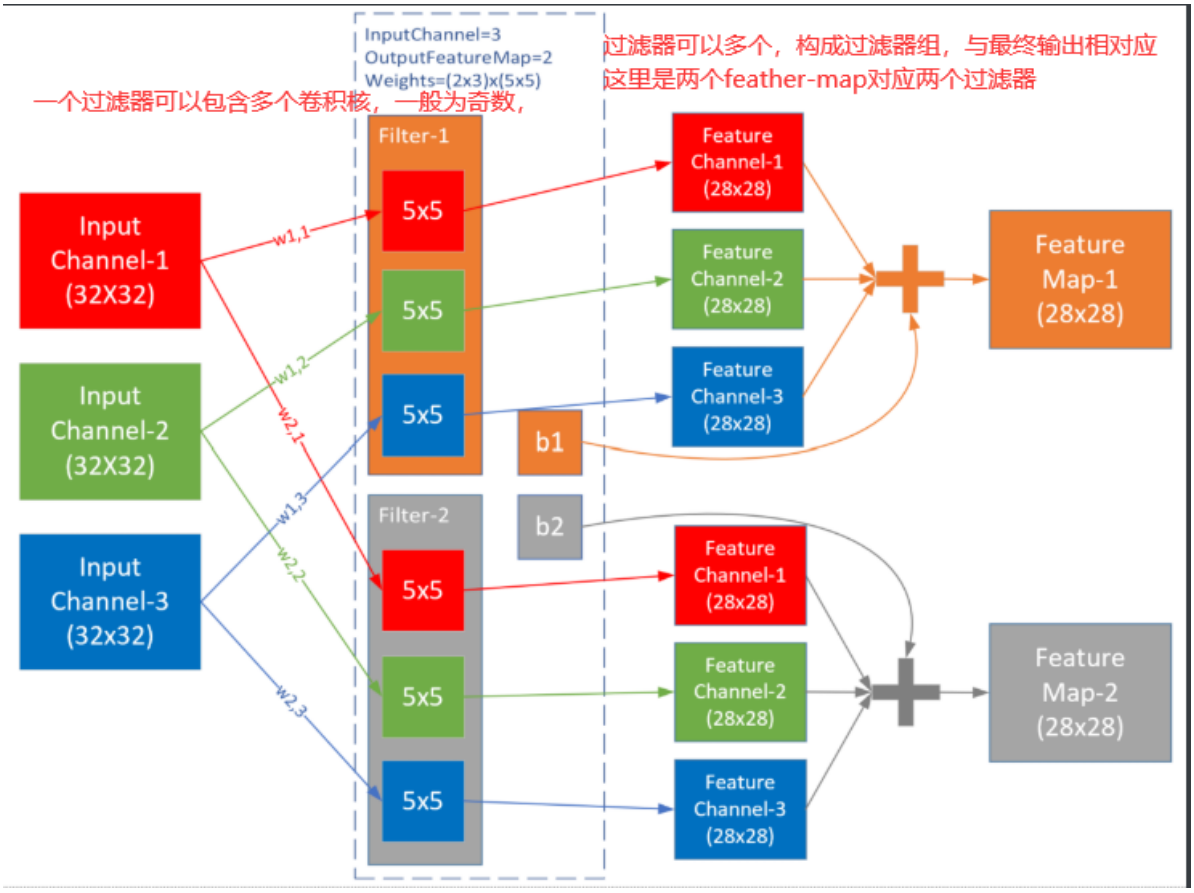
对于灰色的过滤器Filter-2也是一样，先生成三个Feature-2,n，然后相加再加b2，最后得到Result-2。

之所以Feature-m,n还用红绿蓝三色表示，是因为在此时，它们还保留着红绿蓝三种色彩的各自的信息，一旦相加后得到Result，这种信息就丢失了。

1.2.6 卷积的编程模型

上面的偏重于计算，下面的更关注于解释概念之间的关系

- 输入 Input Channel
- 卷积核组 WeightsBias
- 过滤器 Filter
- 卷积核 kernal
- 输出 Feature Map



输入是三维数据（3x32x32），经过2x3x5x5（从大往小的方式看，最外层的包含下去的顺序）的卷积后，输出为三维（2x28x28），维数并没有变化，只是每一维内部的尺寸有了变化，一般都是要向更小的尺寸变化，以便于简化计算。

对于三维卷积，有以下特点：

1. 预先定义输出的feature map的数量，而不是根据前向计算自动计算出来，此例中为2，这样就会有两组WeightsBias
2. 对于每个输出，都有一个对应的过滤器Filter，此例中Feature Map-1对应Filter-1
3. 每个Filter内都有一个或多个卷积核Kernal，对应每个输入通道(Input Channel)，此例为3，对应输入的红绿蓝三个通道，（对每一个颜色淡出处理）
4. 每个Filter只有一个Bias值，Filter-1对应b1，Filter-2对应b2
5. 卷积核Kernal的大小一般是奇数如：1x1, 3x3, 5x5, 7x7等，此例为5x5

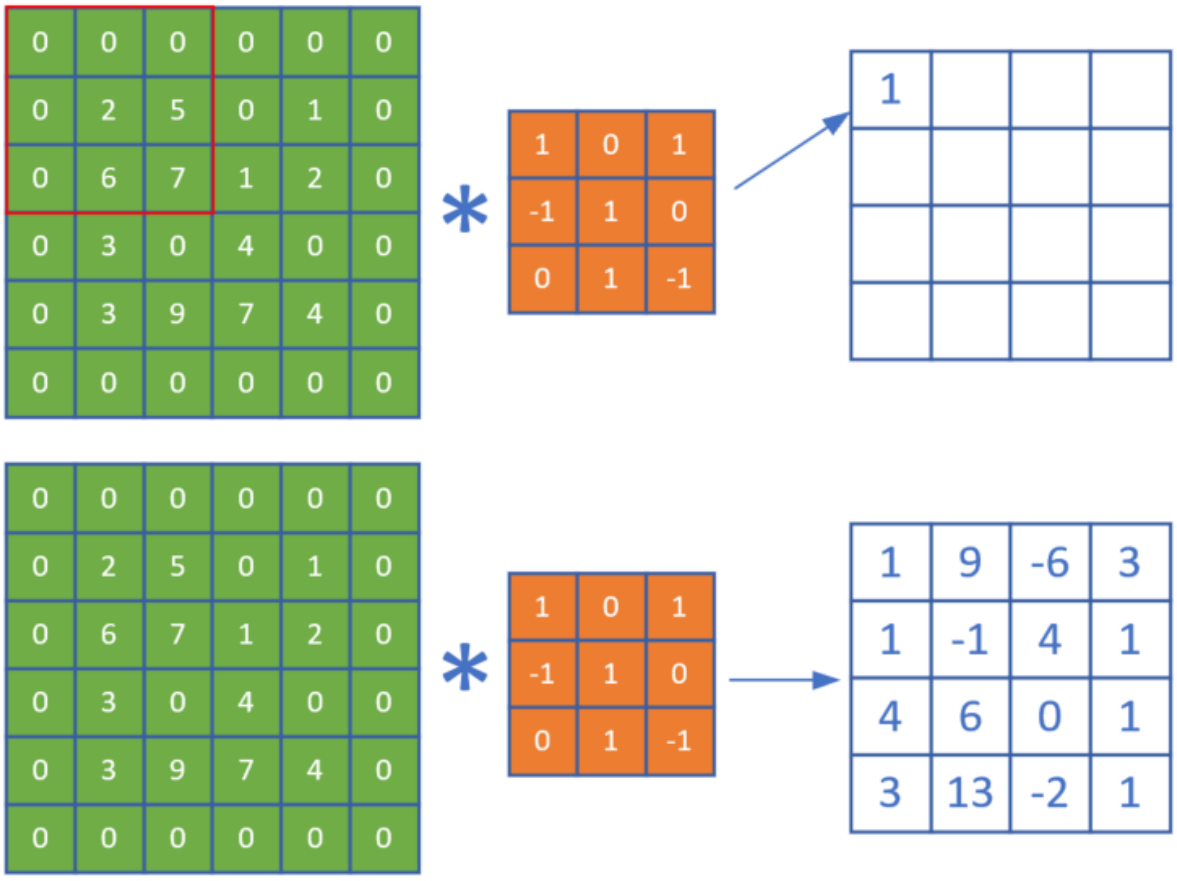
对于上图，我们可以用在全连接神经网络中的学到的知识来理解：

1. 每个Input Channel就是特征输入，在上图中是3个
2. 卷积层的卷积核相当于隐层的神经元，上图中隐层有2个神经元
3. $W(m,n), m=[1,2], n=[1,3]$相当于隐层的权重矩阵$w_{11}, w_{12}, \ldots\ldots$
4. 每个卷积核（神经元）有1个偏移值

1.2.7 填充padding

如果原始图为4x4，用3x3的卷积核进行卷积后，目标图片变成了2x2。如果我们想保持目标图片和原始图片为同样大小，该怎么办呢？一般我们会向原始图片周围填充一圈0，然后再做卷积。

如果原始图为4x4，用3x3的卷积核进行卷积后，目标图片变成了2x2。如果我们想保持目标图片和原始图片为同样大小，该怎么办呢？一般我们会向原始图片周围填充一圈0，然后再做卷积。如下图：



两点注意：

1. 一般情况下，我们用正方形的卷积核，且为奇数
2. 如果计算出的输出图片尺寸为小数，则取整，不做四舍五入

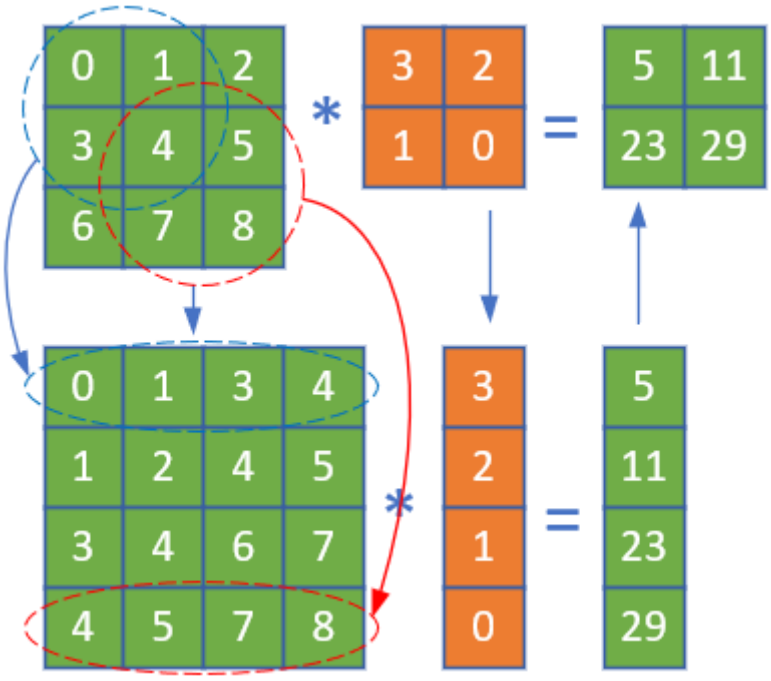
1.3 卷积的向前计算

计算的方式：

1. 直接利用循环进行执行
2. 利用开源项目numba，进行速度优化
3. 卷积操作转换为矩阵操作（im2col函数）

1.3.1 卷积操作转换为矩阵操作的原理

图示如下：



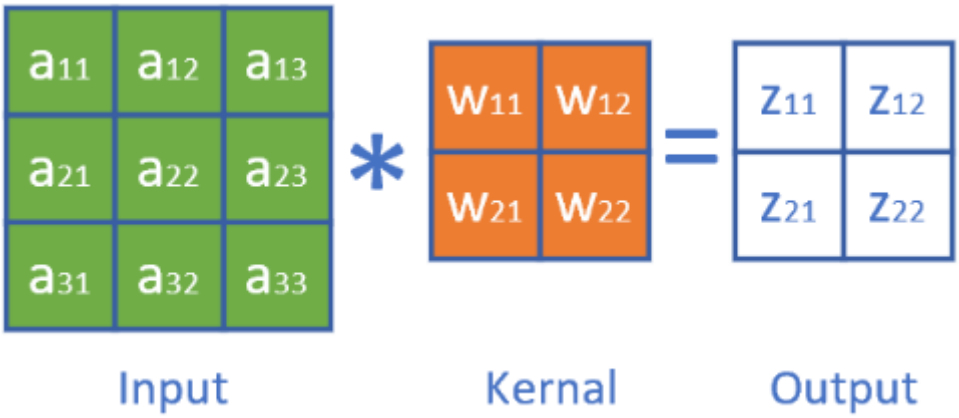
1.4 卷积的反向传播

1.4.1 原理

同全连接层一样，卷积层的训练也需要从上一层回传的误差矩阵，然后计算：

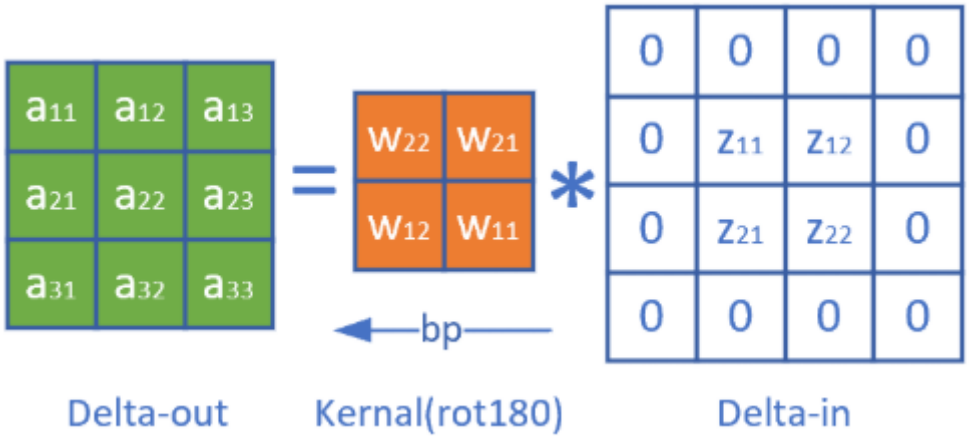
1. 本层的权重矩阵的误差项
2. 本层的需要回传到下一层的误差矩阵

我们回忆，做前向计算就是



那么根据推导，其实反向传播就类似于：

因此，我们把传入的误差矩阵Delta-In做一个zero padding，再乘以旋转180度的卷积核，就是要传出的误差矩阵Delta-Out



最后形成一个简洁的公式：

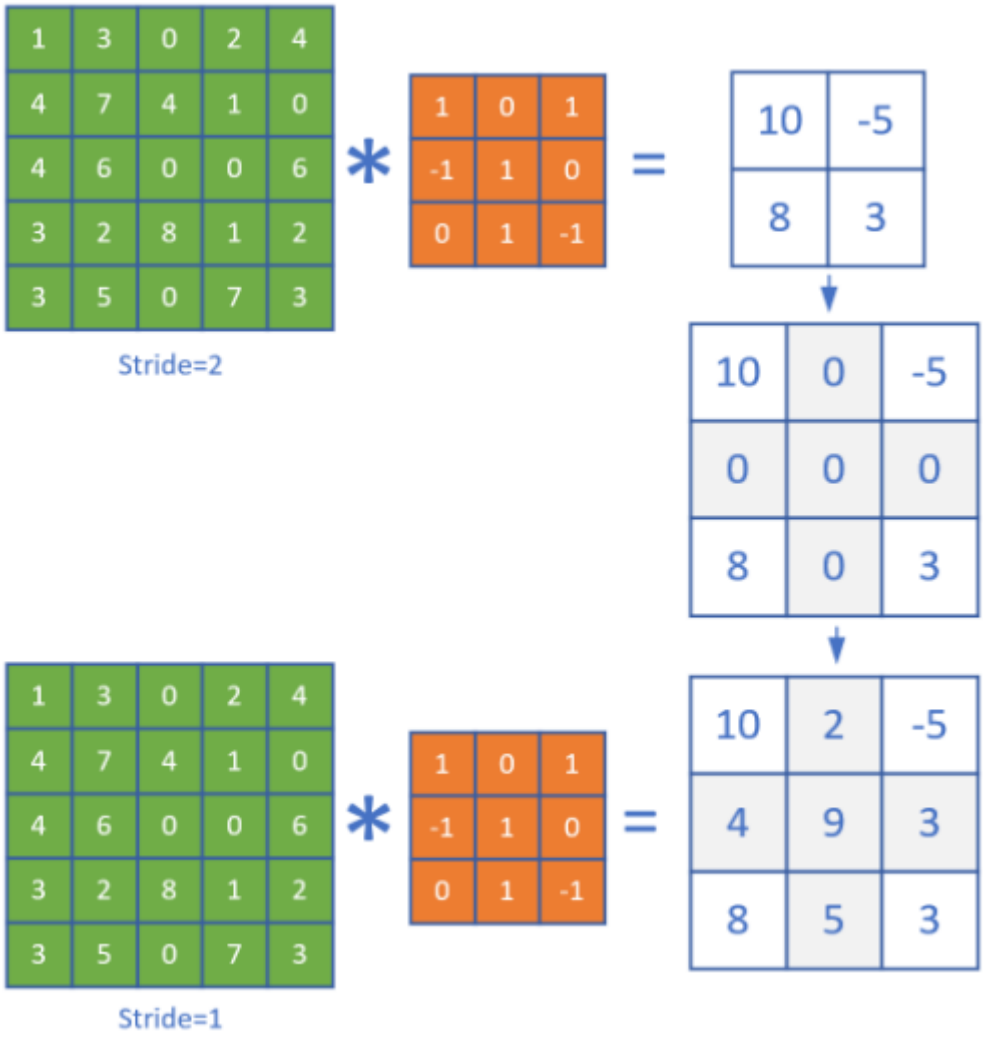
最后可以统一成为一个简洁的公式：

$$\delta_{out} = \delta_{in} * W^{rot180} \qquad (8)$$

并且可以推导得出：做填充的时候，具体的填充数目满足：

当Weights是$N \times N$时，δ_{in}需要padding=N-1，即加N-1圈0

1.4.2 步长不为1时的梯度矩阵还原



如果步长不为1，我们可以利用矩阵的计算进行还原，简称就是：

步长为1，就是一个十字

步长为2，就是一个双十字

以此类推：

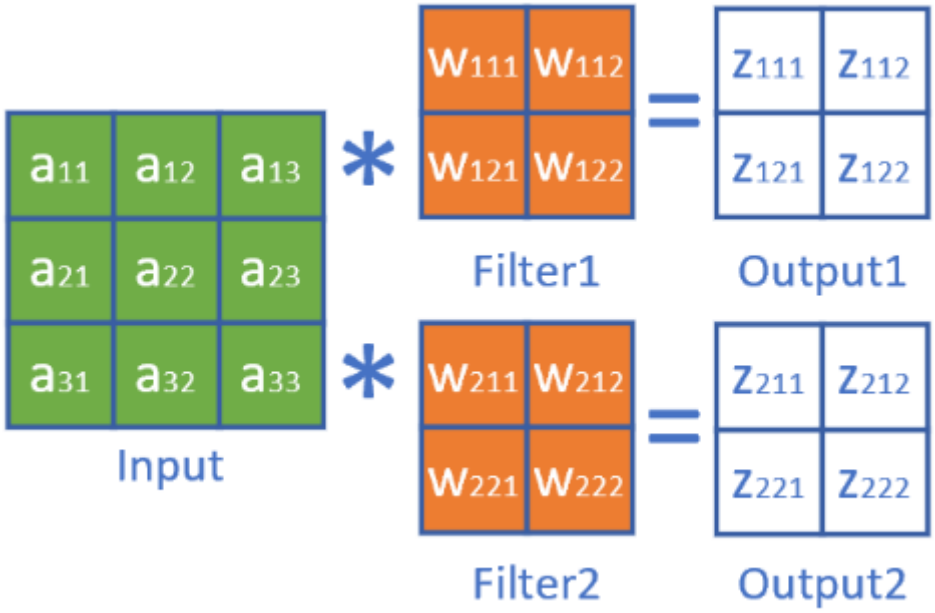
步长为2时，用实例表示就是这样：

$$\begin{bmatrix} \delta_{11} & 0 & \delta_{12} & 0 & \delta_{13} \\ 0 & 0 & 0 & 0 & 0 \\ \delta_{21} & 0 & \delta_{22} & 0 & \delta_{23} \end{bmatrix}$$

步长为3时，用实例表示就是这样：

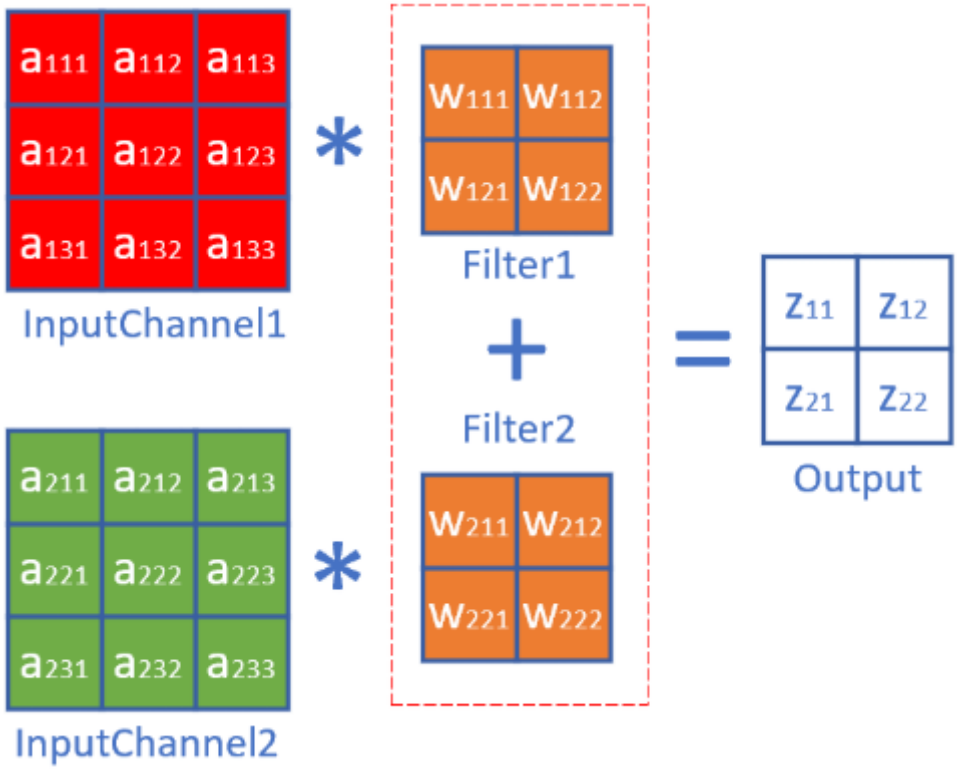
$$\begin{bmatrix} \delta_{11} & 0 & 0 & \delta_{12} & 0 & 0 & \delta_{13} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \delta_{21} & 0 & 0 & \delta_{22} & 0 & 0 & \delta_{23} \end{bmatrix}$$

1.4.3 有多个卷积核时的梯度计算



1.4.4 有多个输入时的梯度计算

当输入层是多个图层时，每个图层必须对应一个卷积核



每个卷积核W可能会有多个filter，或者叫子核，但是一个卷积核只有一个偏移，无论有多少子核。

1.4.5 计算卷积核梯度的实例说明

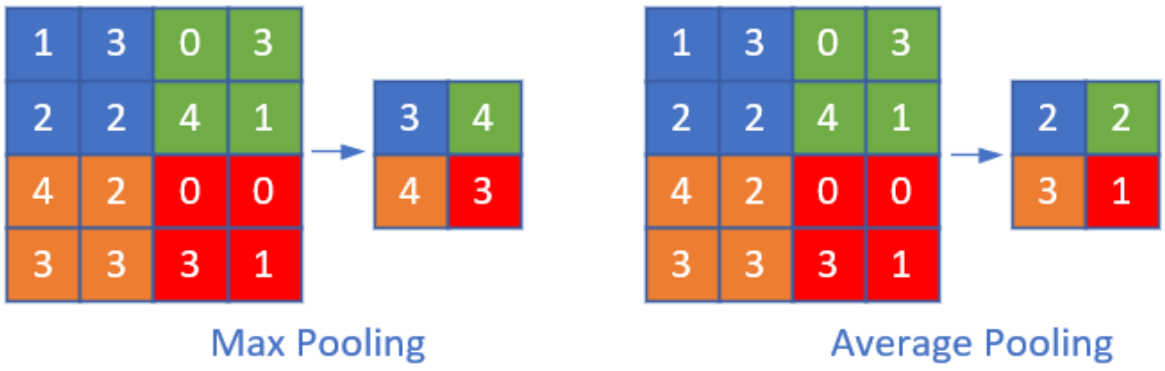
	样本数据	标签数据	预测数据	公式	损失函数
直线拟合	样本点x	标签值y	预测直线z	$z = x \cdot w + b$	均方差
图片拟合	原始图片x	目标图片y	预测图片z	$z = x * w + b$	均方差

直线拟合中的均方差，是计算预测值与样本点之间的距离；图片拟合中的均方差，可以直接计算两张图片对应的像素点之间的差值。

1.5 池化层

池化 pooling，又称为下采样，downstream sampling or sub-sampling。

1.5.1 池化的方式



1.5.1.1 大值池化 Max Pooling

最大值池化，是取当前池化视野中所有元素的最大值，输出到下一层特征图中。

1.5.1.2 平均值池化 Mean/Average Pooling

平均值池化，是取当前池化视野中所有元素的平均值，输出到下一层特征图中。

1.5.2 池化的目的

- 扩大视野：就如同先从近处看一张图片，然后离远一些再看同一张图片，有些细节就会被忽略
- 降维：在保留图片局部特征的前提下，使得图片更小，更易于计算
- 平移不变性，轻微扰动不会影响输出：比如上图中最大值池化的4，即使向右偏一个像素，其输出值仍为4
- 维持同尺寸图片，便于后端处理：假设输入的图片不是一样大小的，就需要用池化来转换成同尺寸图片

1.5.3 池化的使用方式

常用的模式： 池化的尺寸和步长值尽量的相等

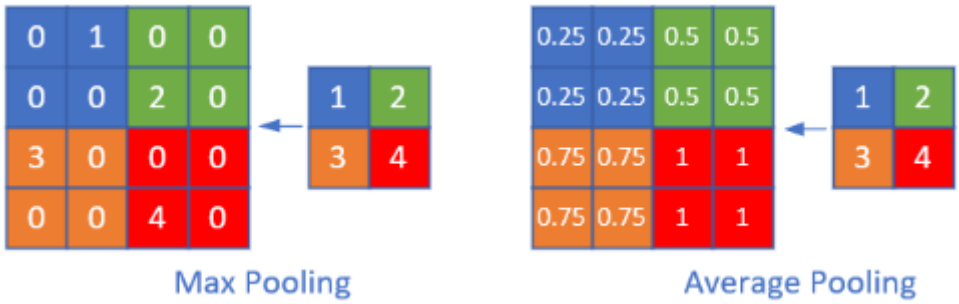
池化层不会改变图片的深度，即D（图层）值前后相同。

1.5.4 池化的训练

如下; $[[1,2],[3,4]]$是上一层网络

回传的残差，那么：

- 对于最大值池化，残差值会回传到当初最大值的位置上，而其它三个位置的残差都是0。
- 对于平均值池化，残差值会平均到原始的4个位置上。

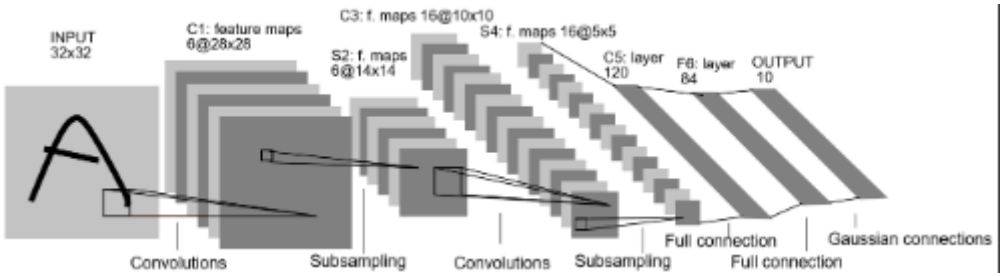


无论是max pooling还是mean pooling，都没有要学习的参数，所以，在卷积网络的训练中，池化层需要做的只是把误差项向后传递，不需要计算任何梯度。

2. 经典的卷积神经网络模型

2.1 介绍

2.1.1 LeNet (1998)

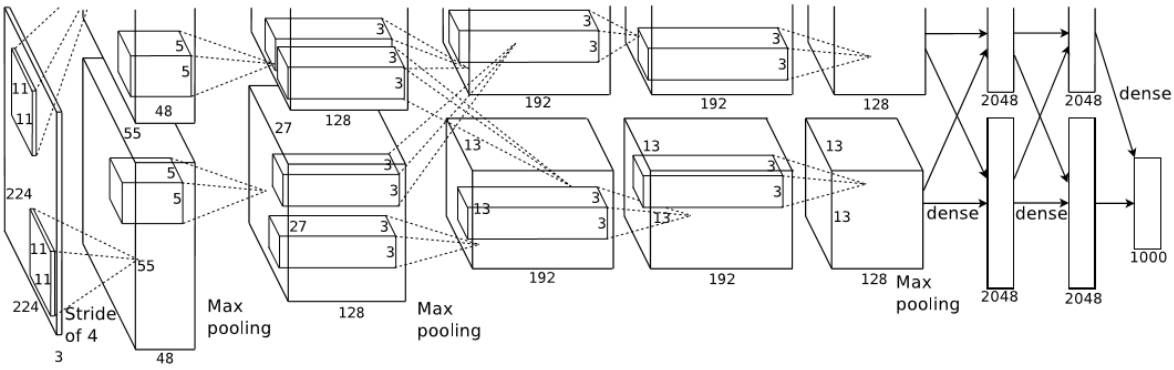


卷积神经网络的最基本的架构就定下来了：卷积层、池化层、全连接层。

如今各大深度学习框架中所使用的LeNet都是简化改进过的LeNet-5（5表示具有5个层），和原始的LeNet有些许不同，比如把激活函数改为了现在很常用的ReLu。LeNet-5跟现有的conv->pool->ReLU的套路不同，它使用的方式是conv1->pool->conv2->pool2再接全连接层，但是不变的是，卷积层后紧接池化层的模式依旧不变。

2.1.2 AlexNet (2012)

AlexNet网络结构在整体上类似于LeNet，都是先卷积然后在全连接。但在细节上有很大不同。AlexNet更为复杂。AlexNet有60 million个参数和65000个神经元，五层卷积，三层全连接网络，最终的输出层是1000通道的Softmax。



AlexNet的特点：

- 比LeNet深和宽的网络

  使用了5层卷积和3层全连接，一共8层。特征数在最宽处达到384。

- 数据增强

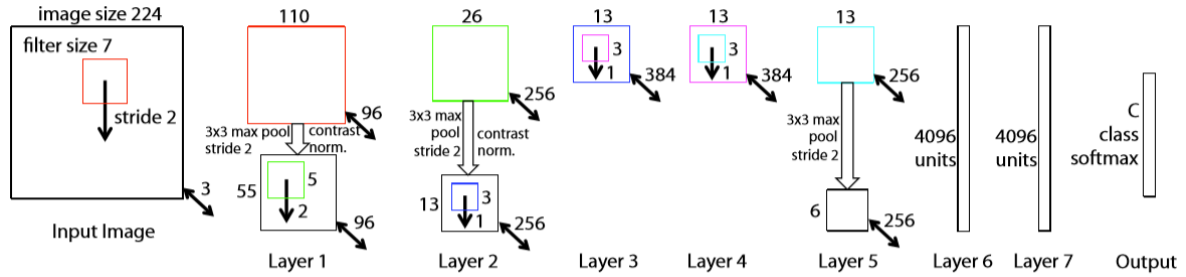
  针对原始图片256x256的数据，做了随机剪裁，得到224x224的图片若干张。

- 使用ReLU做激活函数
- 在全连接层使用DropOut
- 使用LRN

  LRN的全称为Local Response Normalizatio，局部响应归一化，是想对线性输出做一个归一化，避免上下越界

2.1.3 ZFNet (2013)

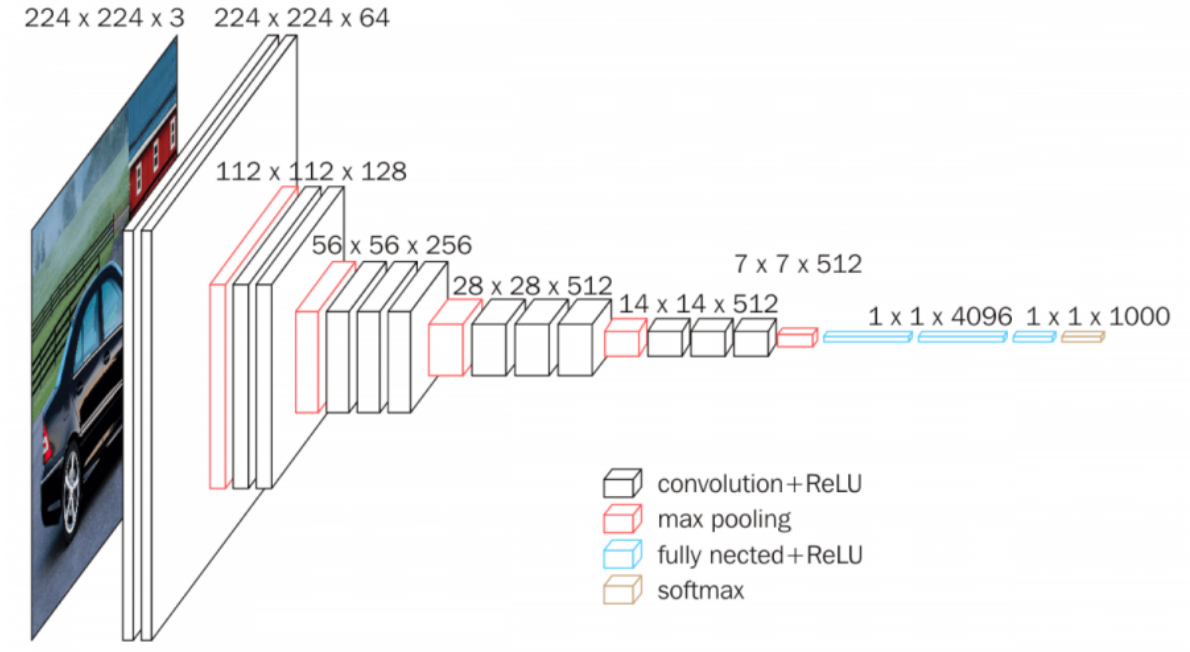
对AlexNet的优化



2.1.4 VGGNet (2015)

常用来做图形特征的提取

下图为VGG16（16层的VGG）模型结构。

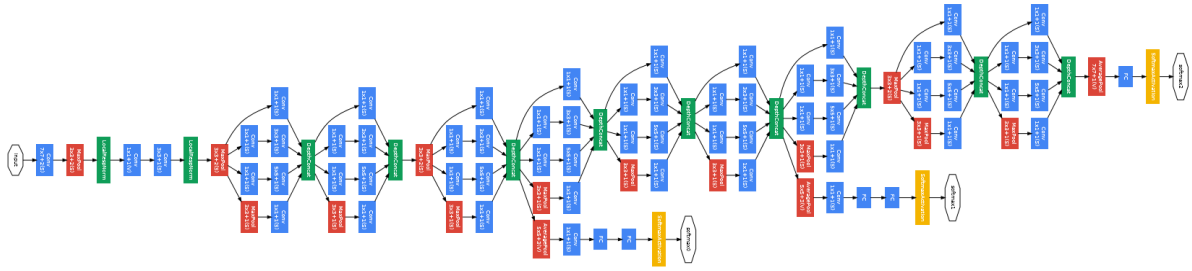


VGGNet的卷积层有一个特点：特征图的空间分辨率单调递减，特征图的通道数单调递增，使得输入图像在维度上流畅地转换到分类向量。用通俗的语言讲，就是特征图尺寸单调递减，特征图数量单调递增。从上面的模型图上来看，立体方块的宽和高逐渐减小，但是厚度逐渐增加。

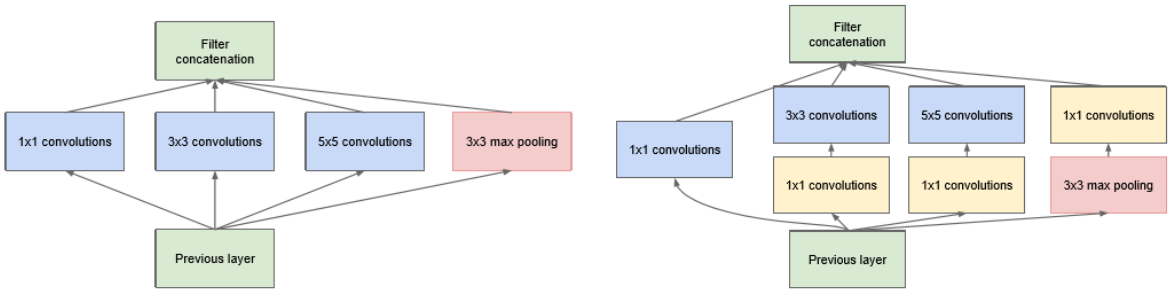
2.1.5 GoogLeNet (2014)

GoogLeNet跟AlexNet,VGG-Nets这种单纯依靠加深网络结构进而改进网络性能的思路不一样，它另辟幽径，在加深网络的同时（22层），也在网络结构上做了创新，引入Inception结构代替了单纯的卷积+激活的传统操作（这思路最早由Network in Network提出）

下图是GoogLeNet的模型结构图。



Inception结构图：蓝色为卷积运算，红色为池化运算，黄色为softmax分类。

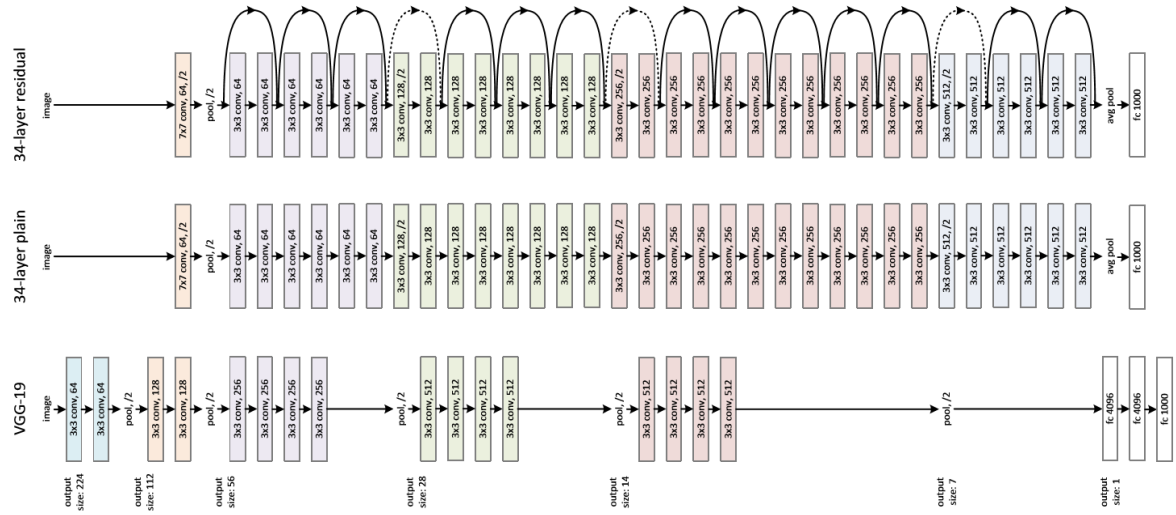


(a) Inception module, naïve version (b) Inception module with dimension reductions

2.1.6 ResNets (2015)

残差网络，

下图是ResNets的模型结构。



若将输入设为X，将某一有参网络层设为H，那么以X为输入的此层的输出将为H(X)。一般的卷积神经网络网络如Alexnet/VGG等会直接通过训练学习出参数函数H的表达，从而直接学习X -> H(X)。而残差学习则是致力于使用多个有参网络层来学习输入、输出之间的参差即H(X) - X即学习X -> (H(X) - X) + X。其中X这一部分为直接的identity mapping，而H(X) - X则为有参网络层要学习的输入输出间残差。

2.1.7 DenseNet (2017)

DenseNet是一种具有密集连接的卷积神经网络。在该网络中，任何两层之间都有直接的连接，也就是说，网络每一层的输入都是前面所有层输出的并集，而该层所学习的特征图也会被直接传给其后面所有层作为输入。

下图是ResNets的模型结构。

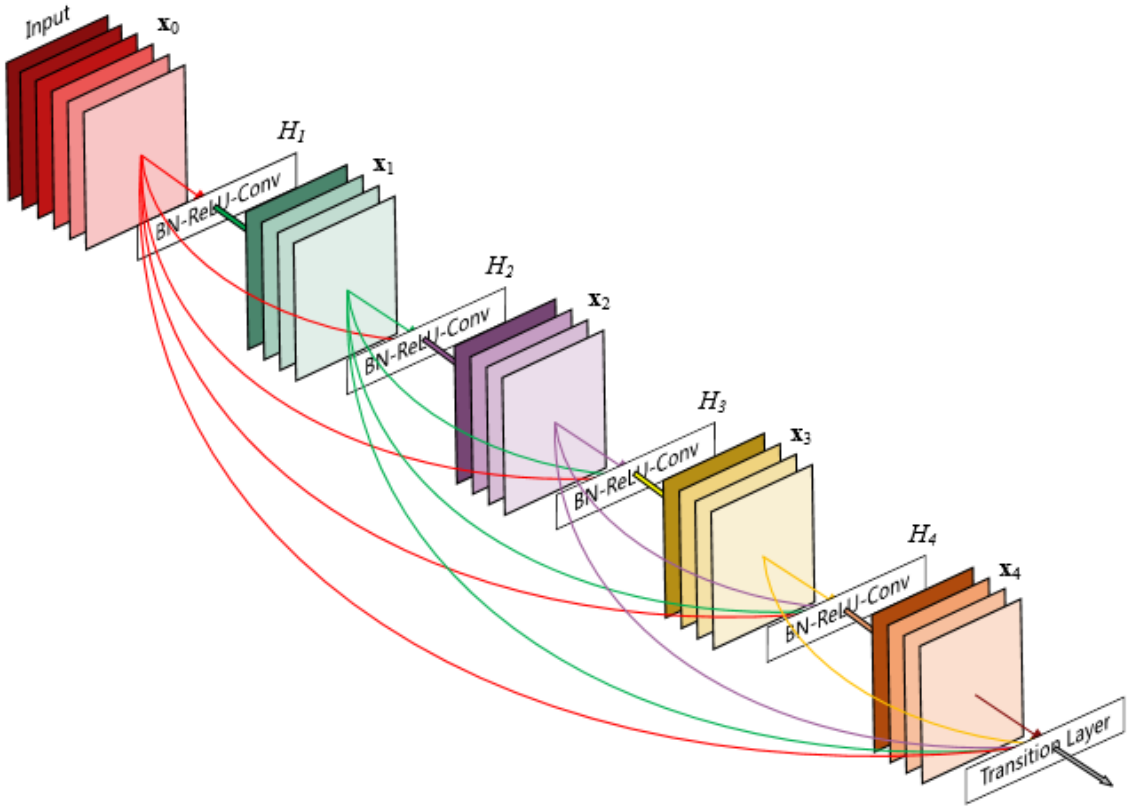


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

1. 相比ResNet拥有更少的参数数量
2. 旁路加强了特征的重用（其多次学习冗余的特征，特征复用是一种更好的特征提取方式）
3. 网络更易于训练,并具有一定的正则效果
4. 缓解了gradient vanishing和model degradation的问题