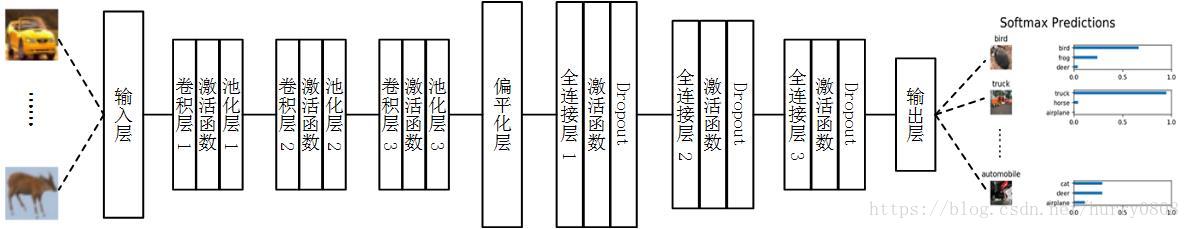
Tensorflow使用数据流图进行数值计算，在使用其构建模型时，先搭建好流图结构——类似于铺好管道，然后再加载数据——向管道中注水，让数据在各节点计算、沿各管道流动；数据在流图中计算、传递时采用多维数组(张量)的形式，因此在Tensorflow中参与计算的均为数组数据类型。

该模型使用Tensorflow构建简单的CNN图像多分类模型，其由3个卷积（涵激活函数）与池化层、扁平层、3个全连接层、一个输出层构成；



# 1.输入层

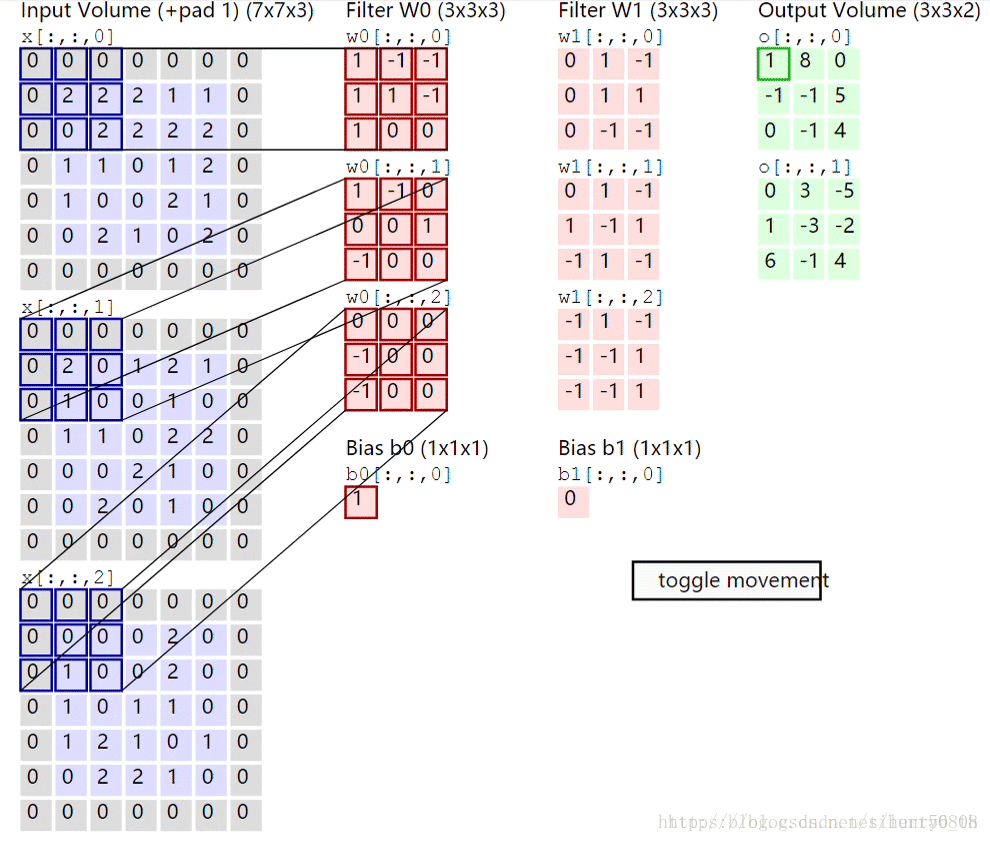
图像数据经过标准化(normalize)、ont-hot编码等预处理后形成张量，本文输入层张量shape为[batch, height, width, channels]。

# 2.卷积层

卷积层与池化层的作用在于：

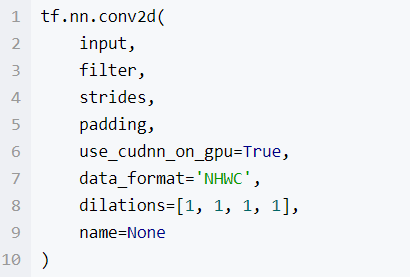
1）. invariance(不变性)，这种不变性包括translation(平移)，rotation(旋转)，scale(尺度)；

2）. 保留主要的特征同时减少参数(降维，效果类似PCA)和计算量，防止过拟合，提高模型泛化能力。



动图网址：<http://cs231n.github.io/convolutional-networks/>

卷积运算函数：



* input：需要做卷积的输入图像，它要求是一个Tensor，shape为[batch, in\_height, in\_width, in\_channels]，具体含义是[训练时一个batch的图片数量, 图片高度, 图片宽度, 图像通道数]，注意这是一个4维的Tensor，要求类型为float32和float64其中之一；
* filter：相当于CNN中的卷积核，它要求是一个Tensor，shape为[filter\_height, filter\_width, in\_channels, out\_channels]，具体含义是[卷积核的高度，卷积核的宽度，图像通道数，卷积核个数]，要求类型与参数input相同，filter的通道数要求与input的in\_channels一致，有一个地方需要注意，第三维in\_channels，就是参数input的第四维；
* strides：卷积时在图像每一维的步长，这是一个一维的向量，长度4，strides[0]=strides[3]=1；
* padding：string类型的量，只能是”SAME”、”VALID”其中之一，这个值决定了不同的卷积方式；卷积运算时卷积核滑动过程中当输入图像(input)的in\_height、in\_width不是filter的filter\_height、filter\_width的整数倍时，”SAME”方式是在input周围填充0以将其in\_height、in\_width补充为filter相应部分的整数倍，”VALID”方式将input的多余部分丢弃，详细介绍请参看[这里](https://blog.csdn.net/jasonzzj/article/details/53930074" \t "_blank)；
* use\_cudnn\_on\_gpu：bool类型，是否使用cudnn加速，默认为true；
* data\_format：指定输入数据、输出数据的格式，取值为”NHWC”、”NCHW”，前者对应于 [batch, height, width, channels]，后者对应于 [batch, channels, height, width]，默认为’NHWC’。
* dilations：一个可选的ints列表；默认为[1,1,1,1]。长度为4的一维张量，每个input维度的膨胀系数； 如果设置为k> 1，则每个该维度上的过滤器元素之间会有k-1个跳过的单元格； 维度顺序由data\_format的值决定； 在Dilations中批次和深度尺寸必须为1；
* name：为该操作指定名称；

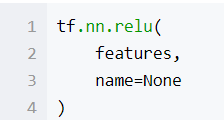
返回值是Tensor，也就是我们常说的feature map。

**Feature map**：在每个卷积层，数据都是以三维形式存在的。你可以把它看成许多个二维图片叠在一起，其中每一个称为一个feature map。在输入层，如果是灰度图片，那就只有一个feature map；如果是彩色图片，一般就是3个feature map（红绿蓝）。层与层之间会有若干个卷积核（kernel），上一层和每个feature map跟每个卷积核做卷积，都会产生下一层的一个feature map。

# 3.激活函数

池化层、全连接层等通过线性运算构建的是线性模型，该模型的表达能力有限；激活函数能够加入非线性因素，从而使模型的表达能力得到极大提升。常用的激活函数有Tanh函数 、ReLU函数 、Leaky ReLU函数 、Maxout函数 等，本文使用ReLU函数作为激活函数。

关于ReLU函数中的参数：

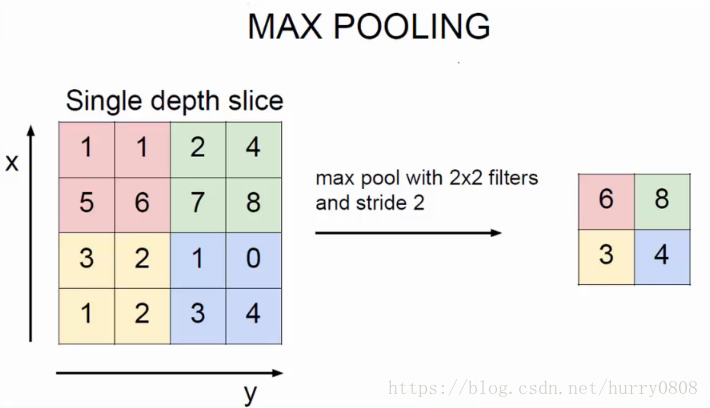


* features: 张量，必须为以下类型中的一种： float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64；
* name: 为该操作指定名称；可为null

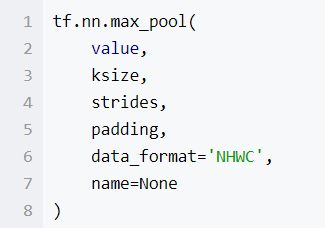
返回值为张量。

# 4.池化层

池化层与卷积层的作用、计算方法类似，也有池化filter、步长、填充方式等参数，所不同的是，池化计算比较简单，常取filter覆盖部分的最大值、或者均值，分别称之为最大池化、平均池化，最大池化的示意图如下所示：



本文用的max\_pool，关于池化层的参数：



参数说明如下：

* value：输入张量， 其格式由data\_format指定；
* ksize：含有4个元素的1D张量，指定池化核的尺寸；
* strides：含有4个元素的1D int 张量，指定池化时在图像每一维的步长，strides[0]=strides[3]=1；
* padding： 边界填充方式，string类型的量，只能是”SAME”、”VALID”其中之一；
* data\_format：数据格式，string类型的量， 只能是’NHWC’、 ‘NCHW’ 、’NCHW\_VECT\_C’ 其中之一；
* name：为该操作指定名称；
* 返回值： Tensor，也就是我们常说的feature map。

上层feature map经池化运算后，其batch、channels均不发生变化，只有height、width会发生变化，输出height、width计算方式同卷积层。

# 5.扁平化层

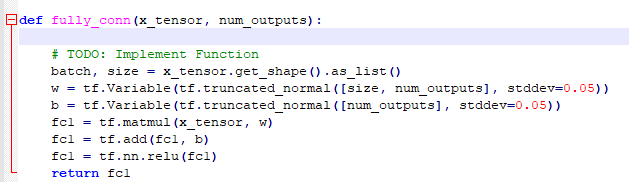
将上一层得到的全部feature map拉直成列向量，作为全连接网络的输入。拉直方式为height\*width\*channels。

图像分类模型训练完成后，本层返回张量的shape(主要是k)就确定了，从而限定了模型预测时输入图像的尺寸——因为在模型确定的情况下，卷积层、池化层等结构、数量均不再改变，预测图像与训练图像尺寸一致时，经卷积、池化等运算后，扁平化输出shape与模型一致。

# 6.全连接层

全连接层在整个卷积神经网络中起到“分类器”的作用、也是模型表示能力的“防火墙”。全连接层的运算形如X\*W+b，X\*W为矩阵乘法，其中X为输入2维数组(shape为[batch\_size, k])，W为权重2维数组(shape为[k，out\_k])，b为偏置，W的第2个元素(out\_k)决定了全连接层运算后输出的2维数组形状(shape为[batch\_size, out\_k])。

由于全连接层参数冗余，需要使用tf.nn.dropout来随机丢弃一些节点，或者使用一些性能优异的网络模型如ResNet和GoogLeNet等来取代全连接层融合学到的深度特征。



# 7.输出层

输出层运算与全连接层类似，只是在设定运算参数时输出节点数量需与分类标记数量相一致，并且在运算完成后再使用tf.nn.softmax函数，得到测试图像属于各分类的概率，该所有概率值之和为1。

# 8.准确率

对于CNN网络输出的分类值的准确率进行显示，

