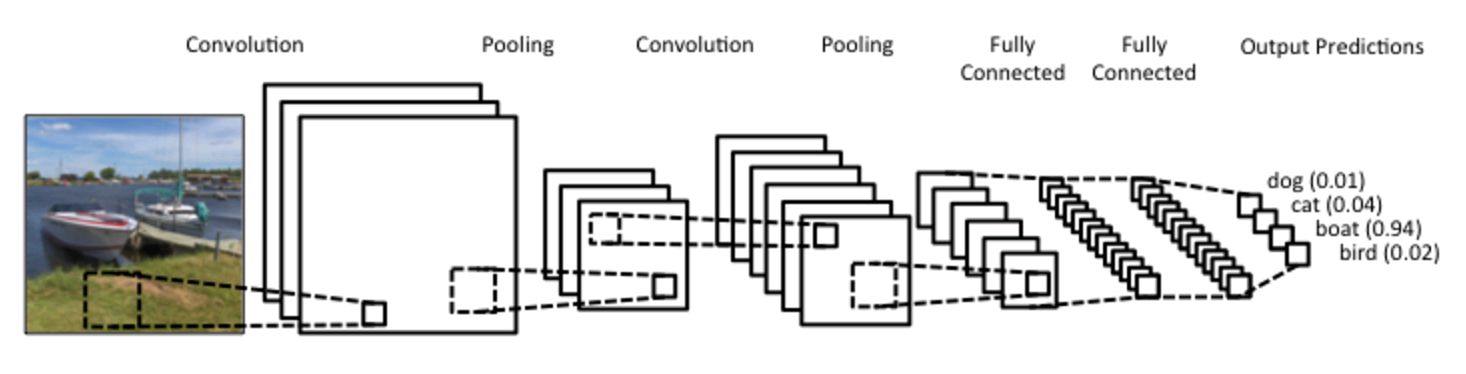
CNN简介

**计算机视觉中的卷积神经网络：**



在图像分类当中 ，CNN可能会从第一层的原始图片中学到边缘检测，然后在第二层通过边缘学习简单的形状，最后用形状检测出人脸这样的高级特征、最后一层就是使用高级特征的分类器。

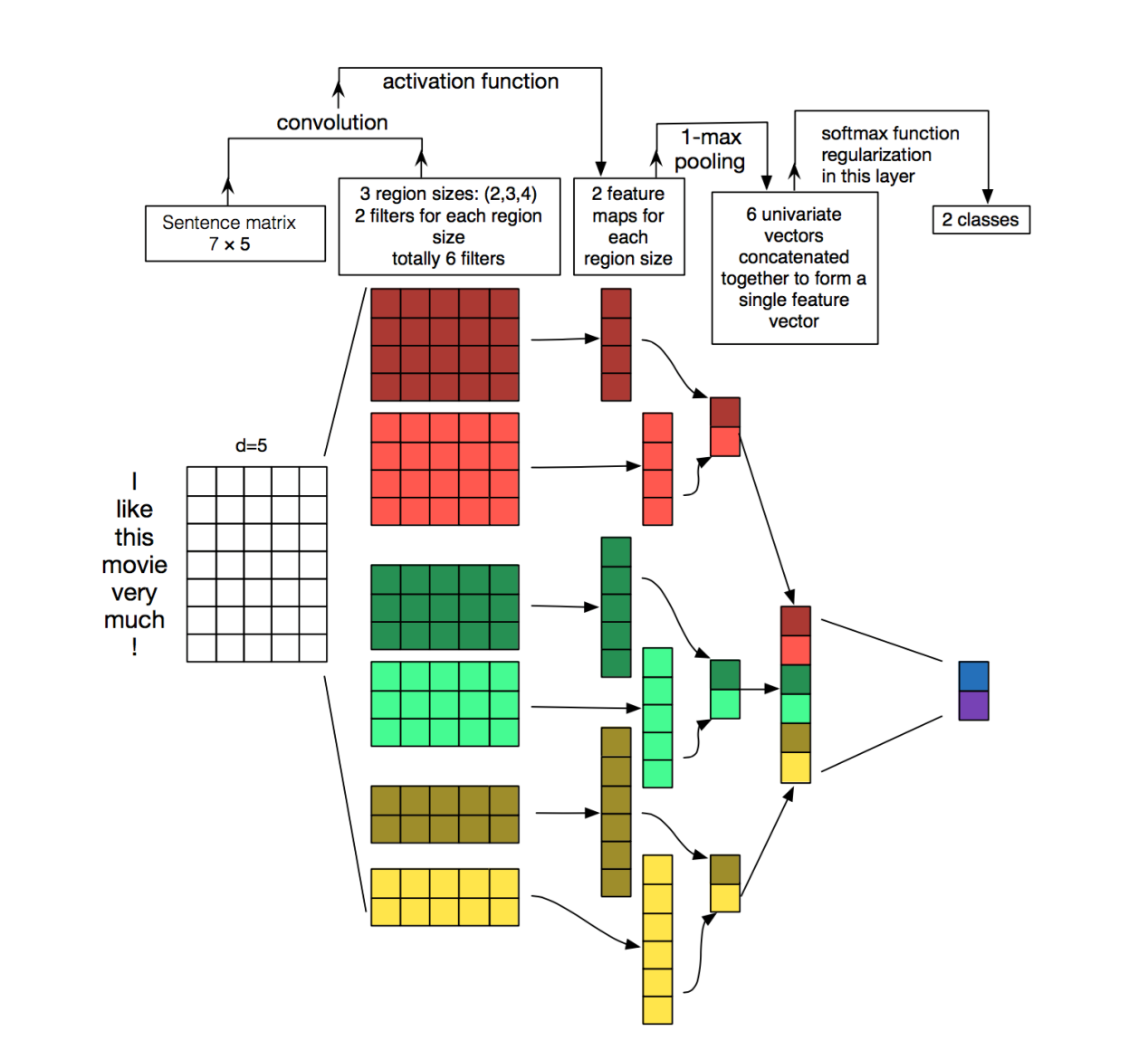
计算过程中还有两个点值得注意：位置不变性(location invariance)和组合性(compositionality)。比方说你想判断图里有没有大象，其实你并不关心大象具体在哪。

第二个关键点就是（局部）组合性。每个卷积核都提取出了一批低级特征，用作之后的高级表达，这也是为什么CNN在计算机视觉上特别好使。就像点动成线，线动成面。

**自然语言处理中的卷积神经网络：**

自然语言处理上面，多数任务的输入都不再是图片像素，而是以矩阵表示的句子或者文档。矩阵的每一行对应一个token，一般是一个单词或者字符。也即每行代表一个词向量，通常是像 [word2vec](https://code.google.com/p/word2vec/)或 [GloVe](http://nlp.stanford.edu/projects/glove/" \t "_blank)词嵌入(word embedding，低维表示)，但这些只能是独热码向量(one-hot vector)，10个单词的句子用100维嵌入，那输入矩阵就是10x100，这就是我们的“图像”。

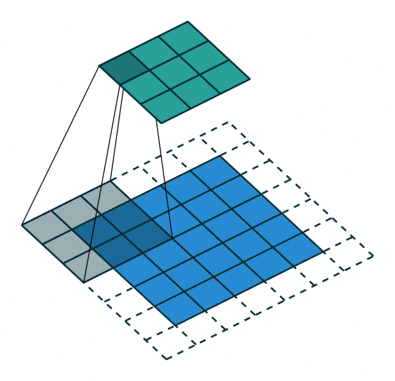
在视觉问题中，卷积核滑过的是图像的一“块”区域，但在自然语言领域里我们一般用卷积核滑过矩阵的一“行”（单词）。然后卷积核的“宽度”就是输入矩阵的宽度，“高度”可能会变，但一般是每次扫过2-5个单词。



**空洞卷积/膨胀卷积：**

Dilation 卷积，也被称为：空洞卷积、膨胀卷积、扩张卷积。

一般的卷积操作：首先，可以通过动态图，理解正常卷积的过程：



如上图，可以看到卷积操作。

对于CNN结构，通常包括如下部分：

输入层 （input layer）---  卷积计算层 （CONV）--- 激励层（RELU） --- 池化层（Pooling） --- 全连接层（FC）

通常利用卷积来实现数据的特征提取。卷积层还有一个权值共享的原则：用一句话表达就是每个神经元只关注一个特征。

当然卷积完经过激励层做一个非线性映射，输出后就到Pooling layer了。

池化层的作用：

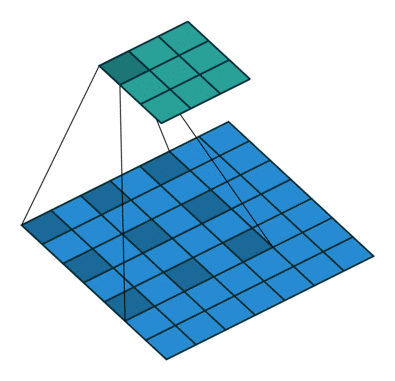
(1)压缩数据和参数的量，减小过拟合。

(2)增大感受野。

主要两种方法：Max Pooling  和  Average Pooling

对于有些算法，池化完还需要upsampling获得原始数的尺寸进行后续操作。由于这种通过卷积操作存在内部数据丢失的问题，存在信息损失，有人提出了dilated conv算法，即不通过池化获得较大的视野，并减小信息损失。

膨胀卷积：

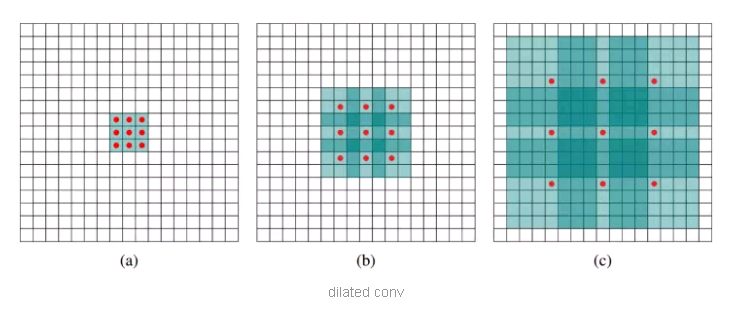


如上图，膨胀卷积的好处是不做pooling损失信息的情况下，加大了感受野，让每个卷积输出都包含较大范围的信息。在图像需要全局信息或者自然语言处理中需要较长的sequence信息依赖的问题中，都能很好的应用。

**IDCNN(Iterated Dilated CNN)：**

模型是4个大的相同结构的Dilated CNN block拼在一起，每个block里面是dilation width为1, 1, 2的三层Dilated卷积层，所以叫做 Iterated Dilated CNN。

**DCNN原理解释：**



(a)图对应3x3的1-dilated conv，和普通的卷积操作一样

(b)图对应3x3的2-dilated conv，实际的卷积kernel size还是3x3，但是空洞为1，也就是对于一个7x7的图像patch，只有9个红色的点和3x3的kernel发生卷积操作，其余的点略过。也可以理解为kernel的size为7x7，但是只有图中的9个点的权重不为0，其余都为0。 可以看到虽然kernel size只有3x3，但是这个卷积的感受野已经增大到了7x7（如果考虑到这个2-dilated conv的前一层是一个1-dilated conv的话，那么每个红点就是1-dilated的卷积输出，所以感受野为3x3，所以1-dilated和2-dilated合起来就能达到7x7的conv）

(c)图是4-dilated conv操作，同理跟在两个1-dilated和2-dilated conv的后面，能达到15x15的感受野。对比传统的conv操作，3层3x3的卷积加起来，stride为1的话，只能达到(kernel-1)\*layer+1=7的感受野，也就是和层数layer成线性关系，而dilated conv的感受野是指数级的增长。

dilated的好处是不做pooling损失信息的情况下，加大了感受野，让每个卷积输出都包含较大范围的信息。

在tensorflow中代码为：

tf.nn.atrous\_conv2d(value,filters,rate,padding,name=None）

除去name参数用以指定该操作的name，与方法有关的一共四个参数：

value：

指需要做卷积的输入图像，要求是一个4维Tensor，具有[batch, height, width, channels]这样的shape，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]

filters：

相当于CNN中的卷积核，要求是一个4维Tensor，具有[filter\_height, filter\_width, channels, out\_channels]这样的shape，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]，同理这里第三维channels，就是参数value的第四维

rate：

要求是一个int型的正数，正常的卷积操作应该会有stride（即卷积核的滑动步长），但是空洞卷积是没有stride参数的，这一点尤其要注意。取而代之，它使用了新的rate参数，那么rate参数有什么用呢？它定义为我们在输入图像上卷积时的采样间隔，你可以理解为卷积核当中穿插了（rate-1）数量的“0”，把原来的卷积核插出了很多“洞洞”，这样做卷积时就相当于对原图像的采样间隔变大了。具体怎么插得，可以看后面更加详细的描述。此时我们很容易得出rate=1时，就没有0插入，此时这个函数就变成了普通卷积。

padding：

string类型的量，只能是”SAME”,”VALID”其中之一，这个值决定了不同边缘填充方式。

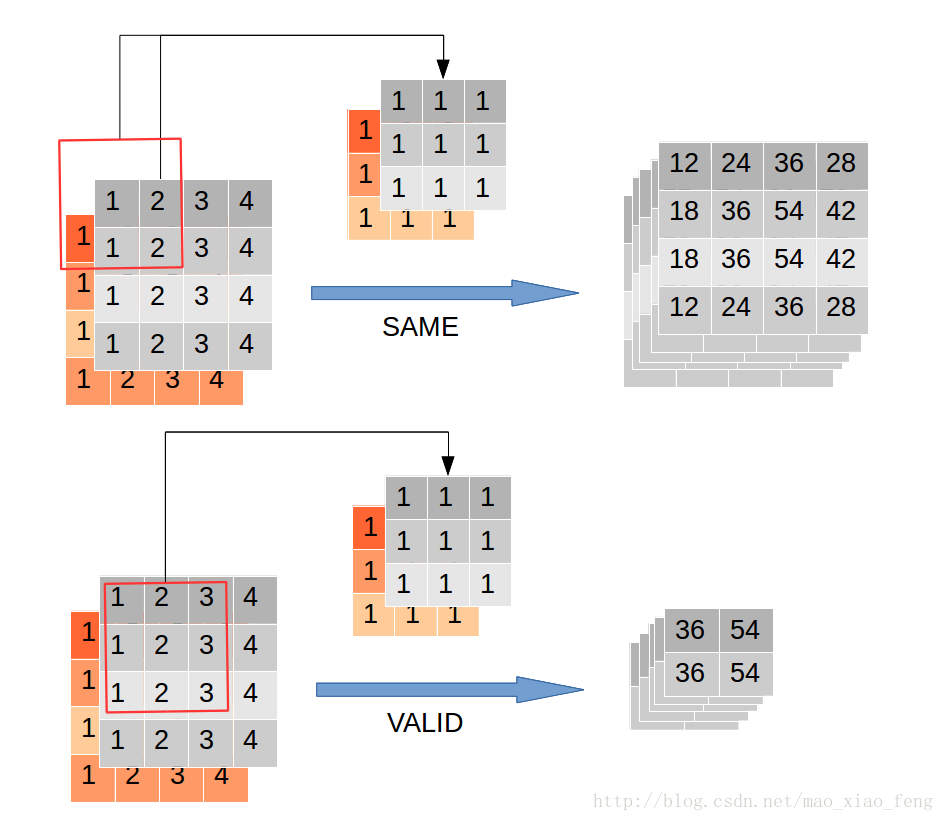
stride”参数：其实这个函数已经默认了stride=1，也就是滑动步长无法改变，固定为1。

结果返回：一个Tensor，填充方式为“VALID”时，返回[batch,height-2\*(filter\_width-1),width-2\*(filter\_height-1),out\_channels]的Tensor，填充方式为“SAME”时，返回[batch, height, width, out\_channels]的Tensor。

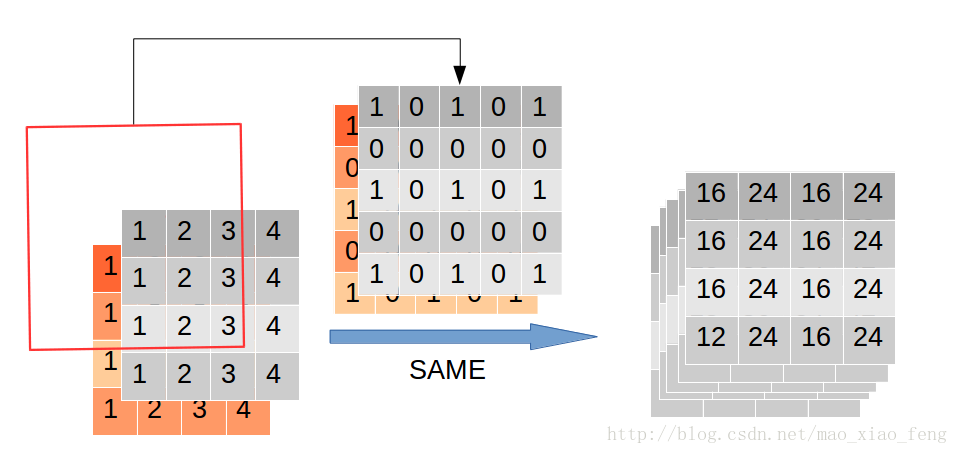
示例：

卷积核：shape=[3,3,2,5]

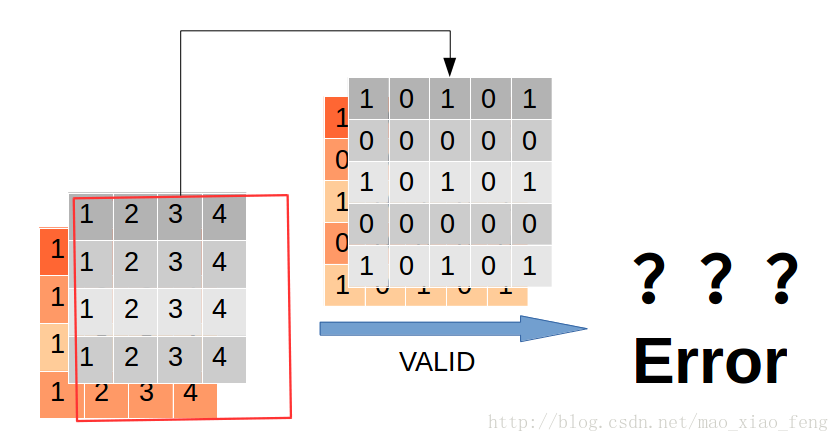
设置rate=1，此时空洞卷积可以看做普通的卷积，分别在SAME和VALID模式下输出如下：



rate=2：



这里我们看到rate=2时，通过穿插“0”，卷积核由3\*3膨胀到了5\*5。再看看“VALID”模式下，会发生什么？



直接报错了，因为卷积核的大小已经超过了原图大小。

参考资料：

<http://www.mamicode.com/info-detail-2339258.html> **膨胀卷积与IDCNN**

<https://www.jianshu.com/p/f743bd9041b3> **空洞卷积(dilated convolution)理解**

[**https://blog.csdn.net/mao\_xiao\_feng/article/details/78003730**](https://blog.csdn.net/mao_xiao_feng/article/details/78003730) **【Tensorflow】tf.nn.atrous\_conv2d如何实现空洞卷积？**

序列标注之IDCNN+CRF

来源：<http://www.crownpku.com//2017/08/26/用IDCNN和CRF做端到端的中文实体识别.html>

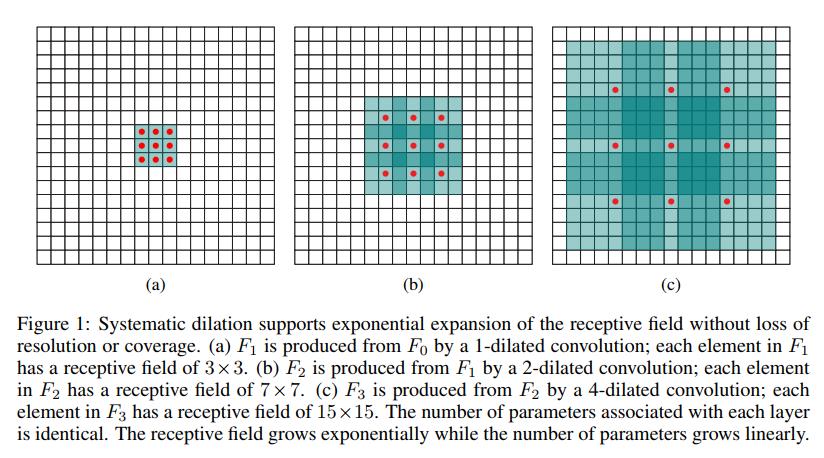
对于序列标注来讲，普通CNN有一个劣势，就是卷积之后，末层神经元可能只是得到了原始输入数据中一小块的信息。而对NER来讲，整个句子的每个字都有可能都会对当前需要标注的字做出影响。为了覆盖到输入的全部信息就需要加入更多的卷积层， 导致层数越来越深，参数越来越多，而为了防止过拟合又要加入更多的Dropout之类的正则化，带来更多的超参数，整个模型变得庞大和难以训练。因为CNN这样的劣势，大部分序列标注问题人们还是使用biLSTM之类的网络结构，尽可能使用网络的记忆力记住全句的信息来对单个字做标注。

但这带来的问题是，biLSTM毕竟是一个序列模型，在对GPU并行计算的优化上面不如CNN那么强大。如何能够像CNN那样给GPU提供一个火力全开的战场，而又像LSTM这样用简单的结构记住尽可能多的输入信息呢？

[Fisher Yu and Vladlen Koltun 2015](https://arxiv.org/abs/1511.07122) 提出了一个dilated CNN的模型，意思是“膨胀的”CNN。想法其实很简单：正常CNN的filter，都是作用在输入矩阵一片连续的位置上，不断sliding做卷积。dilated CNN为这片filter增加了一个dilation width，作用在输入矩阵的时候，会skip掉所有dilation width中间的输入数据；而filter矩阵本身的大小仍然不变，这样filter获取到了更广阔的输入矩阵上的数据，看上去就像是“膨胀”了一般。

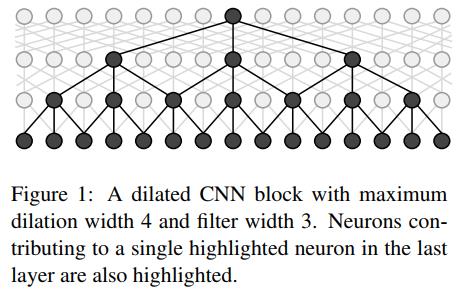
具体使用时，dilated width会随着层数的增加而指数增加。这样随着层数的增加，参数数量是线性增加的，而receptive field却是指数增加的，可以很快覆盖到全部的输入数据。

图片来自 [Fisher Yu and Vladlen Koltun 2015](https://arxiv.org/abs/1511.07122)



对应在文本上，输入是一个一维的向量，每个元素是一个character embedding：

图片来自[Emma Strubell, Patrick Verga, David Belanger, Andrew McCallum 2017](https://arxiv.org/abs/1702.02098)



我们的模型是4个大的相同结构的Dilated CNN block拼在一起，每个block里面是dilation width为1, 1, 2的三层Dilated卷积层，所以叫做 Iterated Dilated CNN。

IDCNN对输入句子的每一个字生成一个logits，这里就和biLSTM模型输出logits之后完全一样，放入CRF Layer，用Viterbi算法解码出标注结果。

在biLSTM或者IDCNN这样的深度网络模型后面接上CRF层是一个序列标注很常见的方法。biLSTM或者IDCNN计算出的是每个词分类的概率，而CRF层引入序列的转移概率，最终计算出loss反馈回网络。网上有不少相关帖子，如[这里](http://www.brucexie.com/2016/11/27/Bi-LSTM%20CRF%20%E6%A8%A1%E5%9E%8B/)