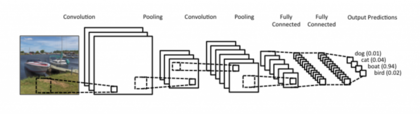
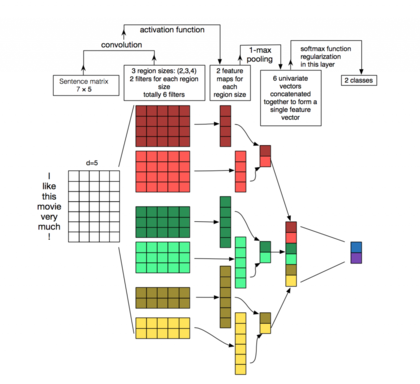
## 卷积神经网络

CNNs本质上就是多层卷积运算，外加对每层的输出用非线性激活函数做转换，比如用[ReLU](https://en.wikipedia.org/wiki/Rectifier_(neural_networks))和[tanh](https://reference.wolfram.com/language/ref/Tanh.html)。在传统的前馈神经网络中，我们把每个输入神经元与下一层的输出神经元相连接。这种方式也被称作是全连接层，或者仿射层。在CNNs中我们不这样做，而是用输入层的卷积结果来计算输出。这相当于是局部连接，每块局部的输入区域与输出的一个神经元相连接。对每一层应用不同的滤波器，往往是如上图所示成百上千个，然后汇总它们的结果。这里也涉及到池化层（降采样），我会在后文做解释。在训练阶段，CNN基于你想完成的任务自动学习滤波器的权重值。举个例子，在图像分类问题中，第一层CNN模型或许能学会从原始像素点检测到一些边缘线条，然后根据边缘线条在第二层检测出一些简单的形状，然后基于这些形状检测出更高级的特征，比如脸部轮廓等。最后一层是利用这些高级特征的一个分类器。



这种计算方式有两点值得我们注意：**位置不变性和组合性**。比如说你想对图片中是否包含大象做分类。因为滤波器是在全图范围内平移，所以并不用关心大象究竟在图片的什么位置。事实上，池化也有助于平移、旋转和缩放的不变性，它对克服缩放因素的效果尤其好。第二个关键因素是（局部）组合性。每个滤波器对一小块局部区域的低级特征组合形成更高级的特征表示。这也是CNNs对计算机视觉作用巨大的原因。我们可以很直观地理解，线条由像素点构成，基本形状又由线条构成，更复杂的物体又源自基本的形状。

在计算机视觉的例子里，我们的滤波器每次只对图像的一小块区域运算，但在处理自然语言时滤波器通常覆盖上下几行（几个词）。因此，滤波器的宽度也就和输入矩阵的宽度相等了。尽管高度，或者区域大小可以随意调整，但一般滑动窗口的覆盖范围是2~5行,综上所述，处理自然语言的卷积神经网络结构是这样的



用于句子分类器的卷积神经网络（CNN）结构示意图。这里我们对滤波器设置了三种尺寸：2、3和4行，每种尺寸各有两种滤波器。每个滤波器对句子矩阵做卷积运算，得到（不同程度的）特征字典。然后对每个特征字典做最大值池化，也就是只记录每个特征字典的最大值。这样，就由六个字典生成了一串单变量特征向量（univariate feature vector），然后这六个特征拼接形成一个特征向量，传给网络的倒数第二层。最后的softmax层以这个特征向量作为输入，用其来对句子做分类；我们假设这里是二分类问题，因此得到两个可能的输出状态

位置不变性和局部组合性对图像来说很直观，但对NLP却并非如此。你也许会很在意一个词在句子中出现的位置。相邻的像素点很有可能是相关联的（都是物体的同一部分），但单词并不总是如此。在很多种语言里，短语之间会被许多其它词所隔离。同样，组合性也不见得明显。单词显然是以某些方式组合的，比如形容词修饰名词，但若是想理解更高级特征真正要表达的含义是什么，并不像计算机视觉那么明显了。

CNNs的主要特点在于速度快。非常的快。卷积运算是计算机图像的核心部分，在GPU级别的硬件层实现。相比于[n-grams](https://en.wikipedia.org/wiki/N-gram)，CNNs表征方式的效率也更胜一筹。由于词典庞大，任何超过3-grams的计算开销就会非常的大。即使Google也最多不超过5-grams。卷积滤波器能自动学习好的表示方式，不需要用整个词表来表征。那么用尺寸大于5行的滤波器完全合情合理了。我个人认为许多在第一层学到的滤波器扑捉到的特征与n-grams非常相似（但不局限），但是以更紧凑的方式表征。

## CNN的超参数

### 窄卷积 vs 宽卷积

在上文中解释卷积运算的时候，我忽略了如何使用滤波器的一个小细节。在矩阵的中部使用3x3的滤波器没有问题，在矩阵的边缘该怎么办呢？左上角的元素没有顶部和左侧相邻的元素，该如何滤波呢？解决的办法是采用补零法（zero-padding）。所有落在矩阵范围之外的元素值都默认为0。这样就可以对输入矩阵的每一个元素做滤波了，输出一个同样大小或是更大的矩阵。补零法又被称为是宽卷积，不使用补零的方法则被称为窄卷积。

当滤波器长度相对输入向量的长度较大时，你会发现宽卷积很有用，或者说很有必要。在上图中，窄卷积输出的长度是 (7-5)+1=3，宽卷积输出的长度是(7+2\*4-5)+1=11。一般形式为IMG_256

### 步长

卷积运算的另一个超参数是步长，即每一次滤波器平移的距离。上面所有例子中的步长都是1，相邻两个滤波器有重叠。步长越大，则用到的滤波器越少，输出的值也越少。

### 池化层

卷积神经网络的一个重要概念就是池化层，一般是在卷积层之后。池化层对输入做降采样。常用的池化做法是对每个滤波器的输出求最大值。我们并不需要对整个矩阵都做池化，可以只对某个窗口区间做池化

在NLP里，我们通常对整个输出做池化，每个滤波器只有一个输出值

### 为什么要池化呢？有许多原因。

池化的特点之一就是它输出一个固定大小的矩阵，这对分类问题很有必要。例如，如果你用了1000个滤波器，并对每个输出使用最大池化，那么无论滤波器的尺寸是多大，也无论输入数据的维度如何变化，你都将得到一个1000维的输出。这让你可以应用不同长度的句子和不同大小的滤波器，但总是得到一个相同维度的输出结果，传入下一层的分类器。

池化还能降低输出结果的维度，（理想情况下）却能保留显著的特征。你可以认为每个滤波器都是检测一种特定的特征，例如，检测句子是否包含诸如“not amazing”等否定意思。如果这个短语在句子中的某个位置出现，那么对应位置的滤波器的输出值将会非常大，而在其它位置的输出值非常小。通过采用取最大值的方式，能将某个特征是否出现在句子中的信息保留下来，但是无法确定它究竟在句子的哪个位置出现。这个信息出现的位置真的很重要吗？确实是的，它有点类似于一组n-grams模型的行为。尽管丢失了关于位置的全局信息（在句子中的大致位置），但是滤波器捕捉到的局部信息却被保留下来了，比如“not amazing”和“amazing not”的意思就大相径庭。

在图像识别领域，池化还能提供平移和旋转不变性。若对某个区域做了池化，即使图像平移/旋转几个像素，得到的输出值也基本一样，因为每次最大值运算得到的结果总是一样的。

### 通道

我们需要了解的最后一个概念是通道。通道即是输入数据的不同“视角”。比如说，做图像识别时一般会用到RGB通道（红绿蓝）。你可以对每个通道做卷积运算，赋予相同或不同的权值。你也同样可以把NLP想象成有许多个通道：把不同类的词向量表征（例如[word2vec](https://code.google.com/p/word2vec/)和[GloVe](http://nlp.stanford.edu/projects/glove/)）看做是独立的通道，或是把不同语言版本的同一句话看作是一个通道。

## 卷积神经网络在自然语言处理的应用

最适合CNNs的莫过于分类任务，如语义分析、垃圾邮件检测和话题分类。卷积运算和池化会丢失局部区域某些单词的顺序信息，因此纯CNN的结构框架不太适用于PoS Tagging和Entity Extraction等顺序标签任务（也不是不可能，你可以尝试输入位置相关的特征）

输入层是一个表示句子的矩阵，每一行是word2vec词向量。接着是由若干个滤波器组成的卷积层，然后是最大池化层，最后是softmax分类器。该论文也尝试了两种不同形式的通道，分别是静态和动态词向量，其中一个通道在训练时动态调整而另一个不变

搭建一个CNN模型结构需要选择许多个超参数，我在上文中已经提到了一些：输入表征（word2vec, GloVe, one-hot），卷积滤波器的数量和尺寸，池化策略（最大值、平均值），以及激活函数（ReLU, tanh）其主要的结论有最大池化效果总是好于平均池化；选择理想的滤波器尺寸很重要，但也根据任务而定需；正则化在NLP任务中的作用并不明显。需要注意的一点是该研究所用文本集里的文本长度都相近，因此若是要处理不同长度的文本，上述结论可能不具有指导意义。