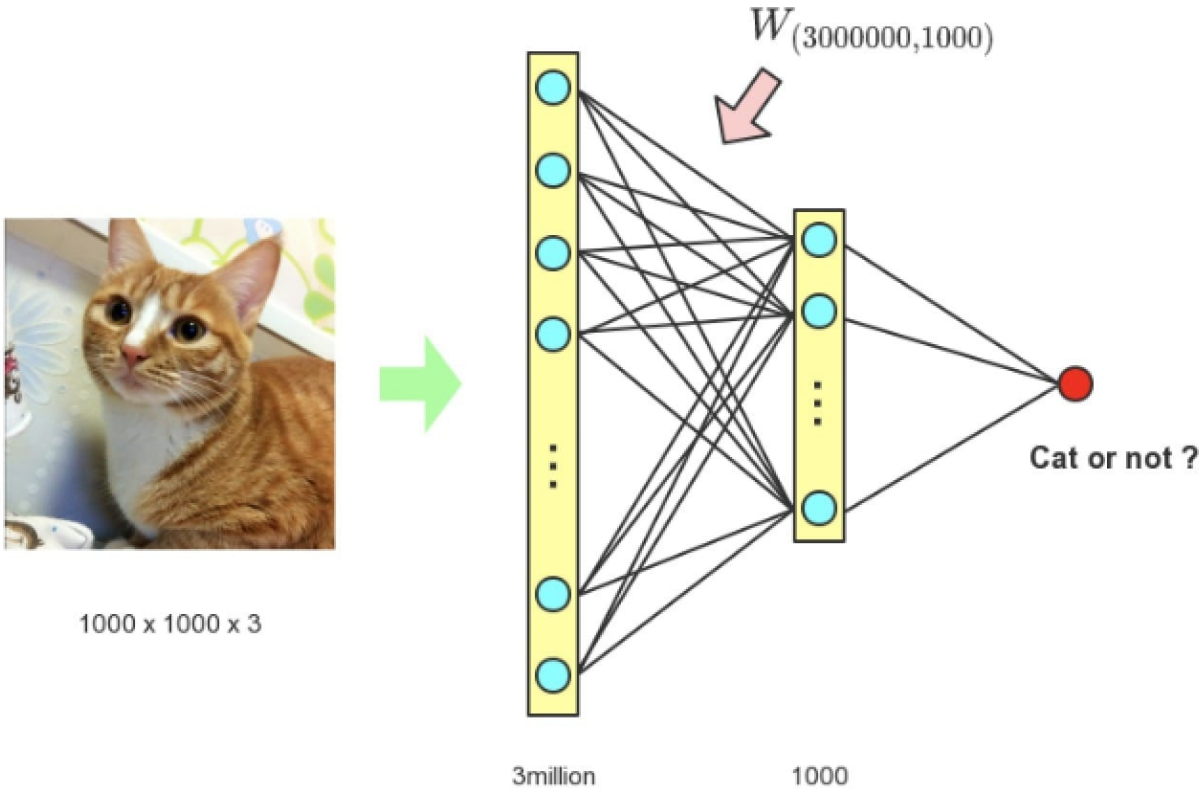
卷积神经网络

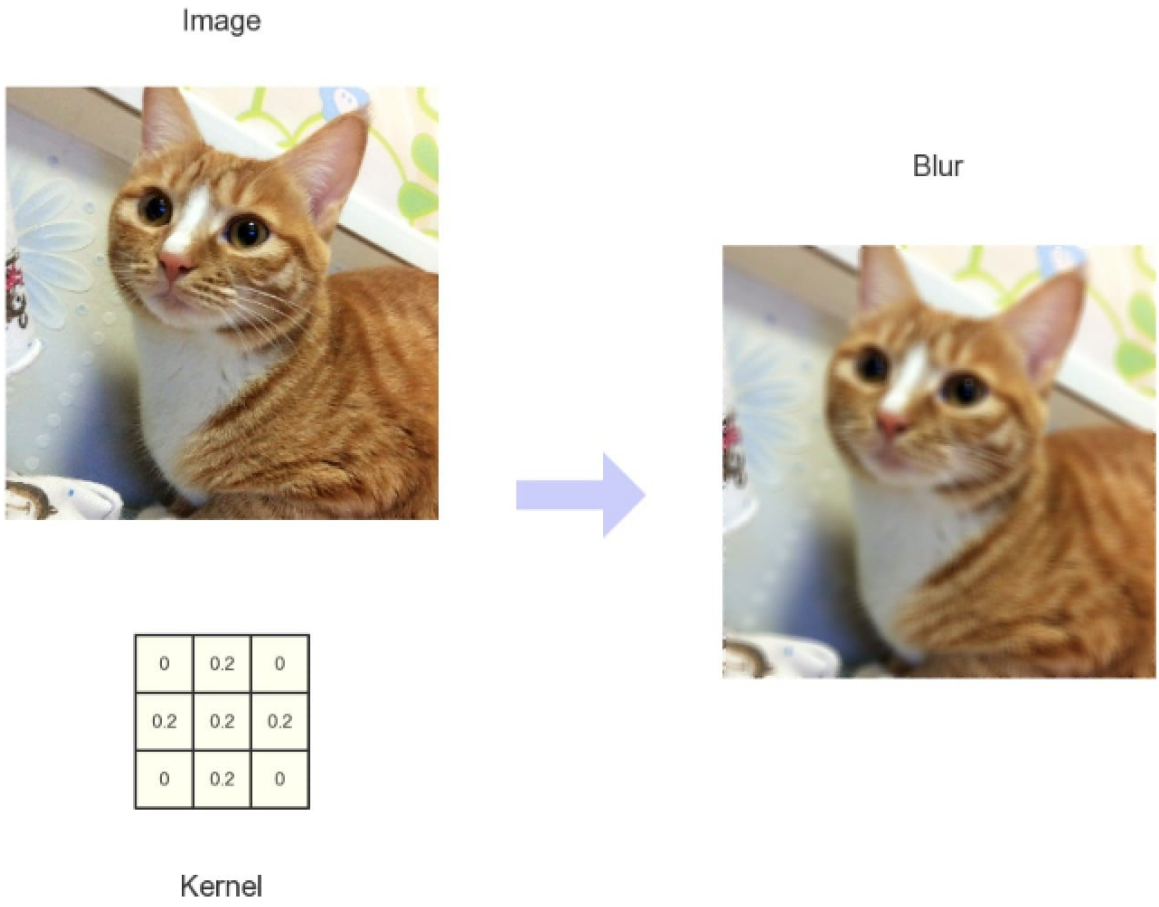
前面我们介绍了回归问题, 分类问题以及深度神经网络, 这些都是以向量为输入来训练一个简单或者复杂的模型. 当我们的输入是图片, 或者说是矩阵的时候, 要怎么样进行处理呢?

有个很简单的方法可以解决这个问题, 就是直接把图片的像素点按照行或者列拉成一个长长的向量, 这样就可以采用之前的方法进行处理了. 这是一个不错的想法, 但图片的大小制约了这个方法实际的操作性.



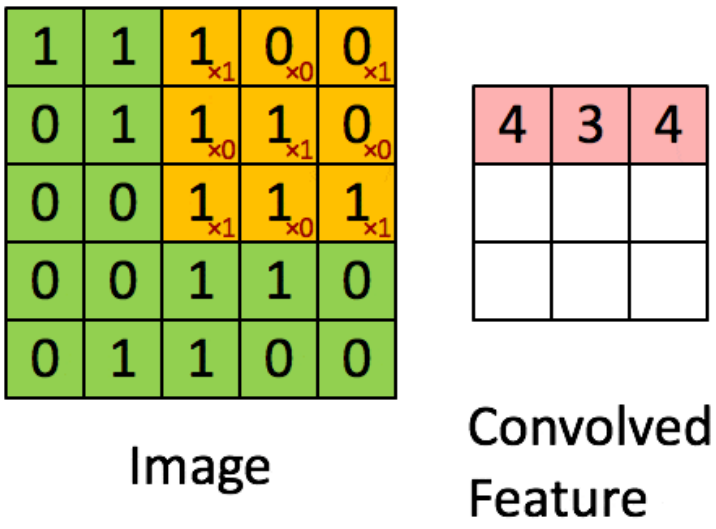
假设我们现在处理一张1000×1000的图片, 考虑到图片具有`R,G,B`三个通道, 那么输入的数据大小将会有`3百万`. 继续, 如果我们希望神经网络的第一个隐藏层具有`1000`个隐藏元, 那么连接输入层和第一个隐藏层的矩阵将会有$3 \times 10^6 \times 1000 = 3 \times 10^9$个参数, 这对于一般机器的存储来说显然是难以接受的.

那么还有一种方法, 人们发现通过传统几何方法对图片进行变换, 可以从中提取某些具有数学意义的特征. 比如在图像几何处理中, 可以对图像进行模糊(`blur`), 锐化(`shapen`), 提取垂直边界, 等等. 我们先看看图像模糊化



上面的图展示了如何将一张图片进行模糊化, 我们来具体解释一下什么是`kernel`以及它是如何产生后面的输出的

传统数字几何处理中, 定义一个3×3的矩阵作为`kernel`, 然后将`kernel`在原图上进行固定步长的滑动, 滑动时对应原图的活动窗口区域, 与这些区域内的点进行点乘运算, 求和, 然后不断的进行滑动, 直到无法继续位置. 这些抽象的描述可以通过下面这张图来具体感受



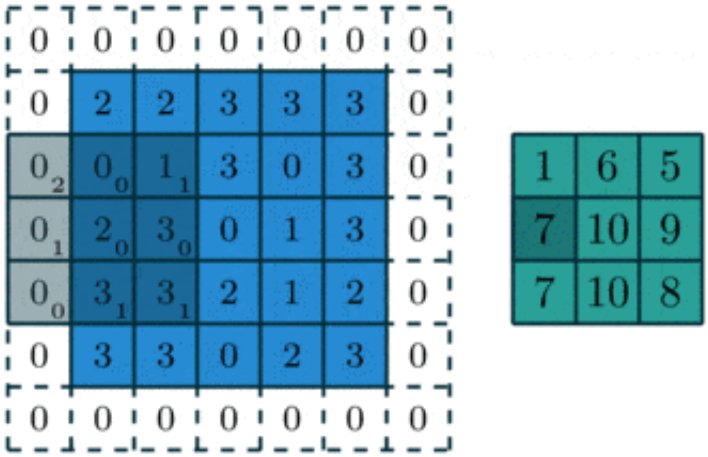
这是类似于一维空间中进行的**卷积运算**, 那么在之后的描述中, 我们称`kernel`为**卷积核**.

在上面图像模糊化的例子中, 我们的卷积核就是一个3×3的矩阵, 它在中心和四个相邻点处的值为`0.2`, 其它点处为`0`. 显然我们可以尝试各种不同的卷积核, 大家可以访问这个博客进行探索, 这里就不进行赘述了

卷积的一些基本知识

卷积运算具有非常多的形式,

- 比如卷积核的大小, 我们称之为`ksize`, 可以是`1, 2, 3, 4 ...`;
- 卷积核一次在原图上移动的步长, 称之为`stride`, 可以是`1, 2, 3, 4, ...`.
- 在实际情况中, 还有一种给原图打补丁的策略, 用来调整输出的大小, 其思想就是在原图的周围填充0, 填充的多少称之为`padding`



在一般的情况中, 固定`ksize`和`stride`时我们可以通过下面的公式计算出输出结果的形状

$$W_{out} = (W - K + 2P)/S + 1 \tag{1}$$

卷积层

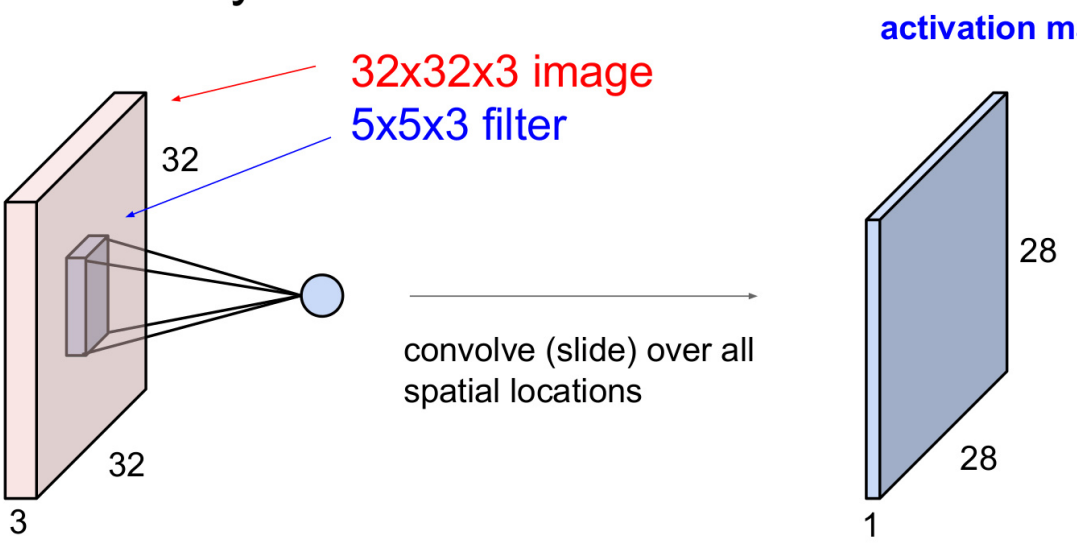
接着, 我们可以思考, 上面的卷积核都是人们探索出来的固定的值, 通常只能解决一些特定的问题.

现在, 我们不妨给卷积核一开始赋以一些随机的数, 把这个卷积核当成神经网络的一部分参数, 对原图像进行一次卷积运算, 然后再连接上一些隐藏层, 最后再到一个输出层, 得到一个损失函数, 通过优化方法去求解所有参数的最优解, 从而自动的找到这个卷积核当中所有元素的值. 这个想法非常好, 其实它就是我们今天的重点:**卷积层**

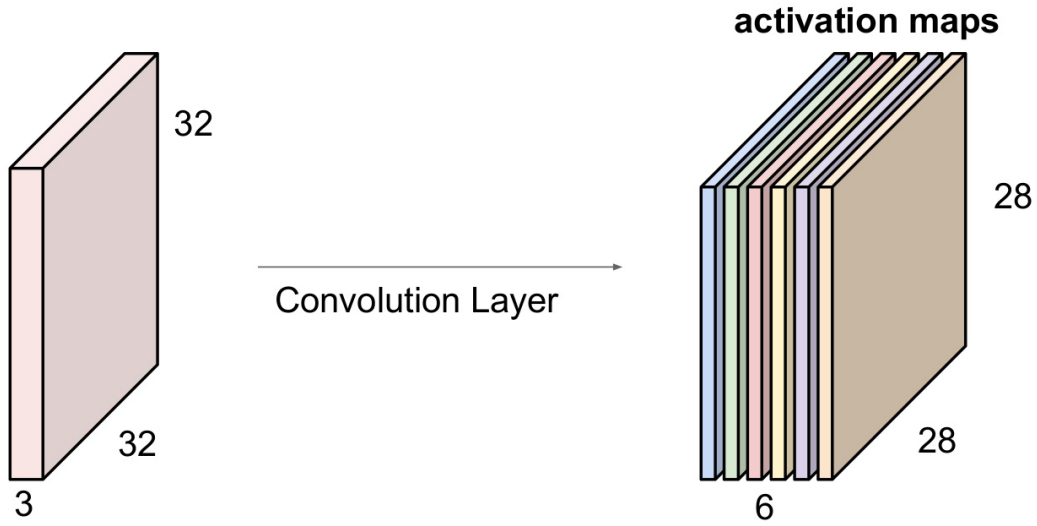
从上面的例子我们可以看出, 一个卷积核就可以得到一个输出, 那么多个卷积核就可以得到多个输出. 因此, 一个卷积层一般由若干个卷积核排列而成.

我们来通过下面这个例子理解卷积层以及它的一些非常重要的细节

Convolution Layer



For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



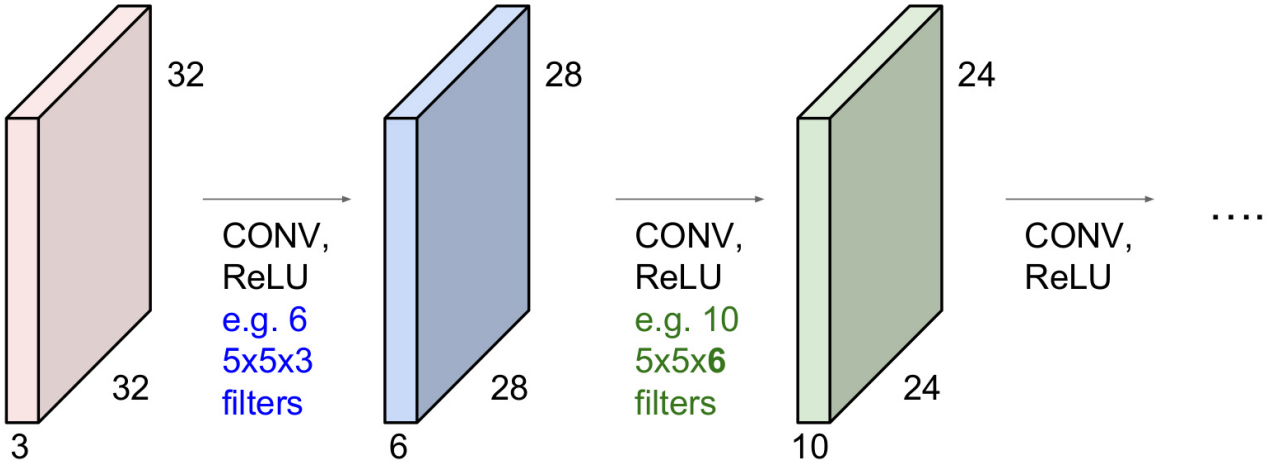
We stack these up to get a "new image" of size 28x28x6!

上面这个图就是一个卷积层的过程, 输入是$32 \times 32 \times 3$的`RGB`通道图片, 通过6个$5 \times 5 \times 3$的卷积核得到$28 \times 28 \times 6$的输出. 现在来解读一下里面出现的数值

- 输入图片大小是3-通道的, 以后我们成为深度为`3`, 为了能够让卷积核和图片进行滑动点乘, 所以卷积核的深度也是`3`, 而卷积核的大小随意, 这里是5×5
- 通过上面计算卷积输出形状的公式, 一个$32 \times 32 \times 3$的图片经过一个$5 \times 5 \times 3$的卷积核得到一个28×28的输出
- 一共有6个卷积核, 所以最后的输出是$28 \times 28 \times 6$的形状

然后, 我们再把卷积层当作一个整体去作为神经网络的一个隐藏层

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



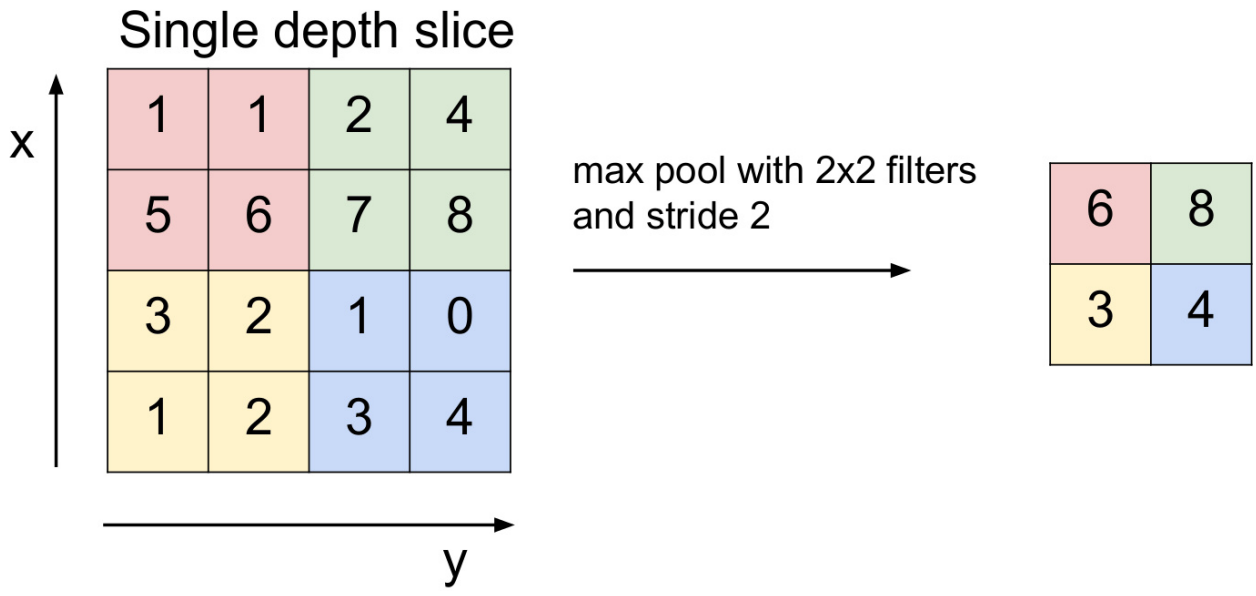
池化层

池化层是卷积神经网络中另外一个非常重要的隐藏层, 它是一个下采样的操作, 用以快速地减小输入的大小同时不至于丢失重要的信息. 现在一般都有两种池化层:

- max pooling(最大值池化层)
- average pooling(平均值池化层)

理解起来也非常简单, 它也有一个矩阵核, 也在原图上进行滑动, 但并没有复杂的点乘相加操作, 而是直接输出活动窗口的统计信息, 比如说最大值池化层输出的是最大值结果, 平均值池化层输出的是平均值的结果

下面这个例子可以帮助我们更直观的理解池化层



卷积神经网络

终于来到了我们今天课程的最终目标: 卷积神经网络. 大家可能更熟悉它的英文名称`CNN`, 是`Convolutional Neural Network`的缩写. 理解了什么是卷积层和池化层, 那么卷积神经网络其实就是`卷积层+池化层+卷积层+池化层+...`这样一个不断循环往复的神经网络结构.

下图是最经典的`CNN`结构, 由Yann LeCun用来解决手写体数字识别的一个网络, 也是第一个成功应用的卷积神经网络, 具有划时代的意义

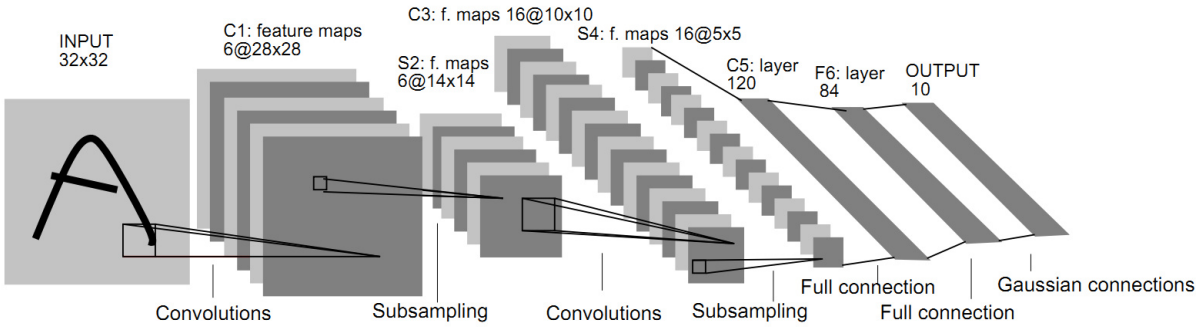


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

大家可以看到, 上面的网络叫做`LeNet-5`, 它具有两个**卷积+池化层**以及三个**全连接层**, 在最后一个池化层之后是通过`flatten`操作将小图片摊平成向量然后连接在一起形成一个长特征向量, 之所以叫全连接层是因为卷积和池化都是局部运算, 而全连接层前后的数据都有权重相连.

结语

这次课程我们了解了什么是卷积层, 池化层以及卷积神经网络. 在之后的课程中, 我们会介绍由上面`LeNet`衍生出来的各种各样的`CNN`以及它们在人工智能这波时代巨浪中扮演的角色