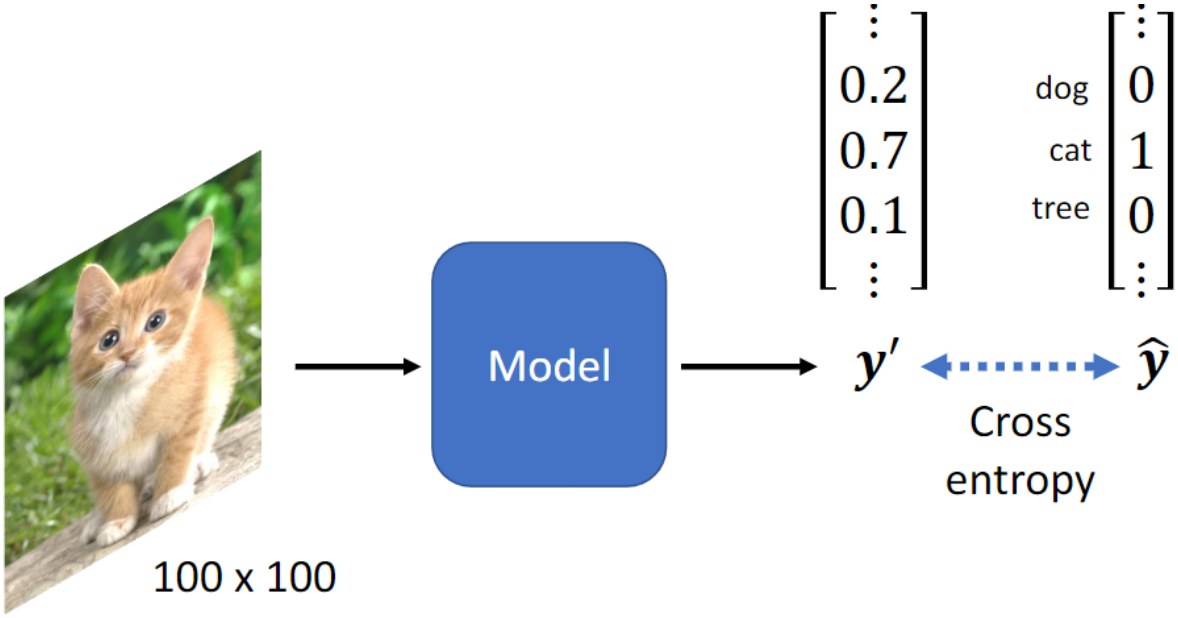
Lecture 3：卷积神经网络（CNN）

> 透过Convolutional Neural Network（CNN）这个例子，来探讨network架构的思想。

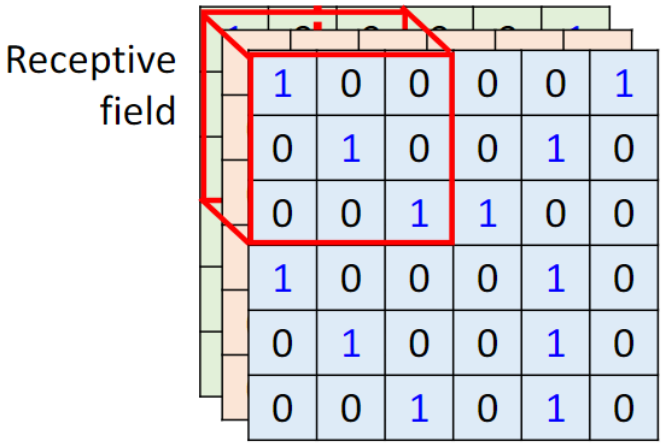
CNN是专门应用在影像数据上的。假设我们现在要做Image Classification任务。满足以下几个前提：输入图像尺寸是一定的（如果有图形不一就先rescale成一样大）；模型的目标是分类，所以每个class（类）表示成一个**one-hot**的向量，最终得到的结果是$\hat{y}$——这个向量的维度代表了分类器可辨识出的种类数。



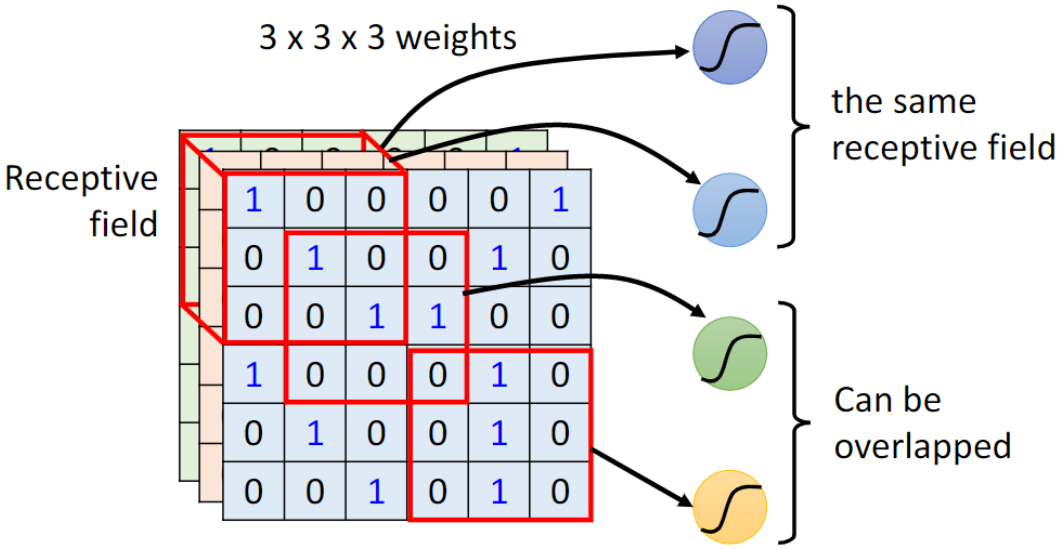
一张图片 = 三维的张量（宽、高、channel）。每隔pixel由（R，G，B）组成。输入向量（=三维的张量），假如宽高为28×28，则输入向量尺度为28×28×3（28×72）。

如果输入第一层network是全连接层，那么参数为28×28×神经元的数量。如此多的参数很容易导致overfitting，而且训练的代价也增大。我们可否减少参数呢？当然可以，首先我们先认知下图像本身的若干特点👇

- *Observation 1*：图像分类一个重要的思想在于为了在图像中识别类别，要去寻找在图像中存在类别相关的特别的pattern，一个neuron并不能看到整张图片，每个神经元看到图像的一部分包含pattern，一起决定判定的结果。事实上，人在观察中也会抓住物件的主要特征来是判别是什么东西。So以下给出这个思想带来的network的变化，简化以尽量避免初始层的全连接层。
  - *Simplification 1*

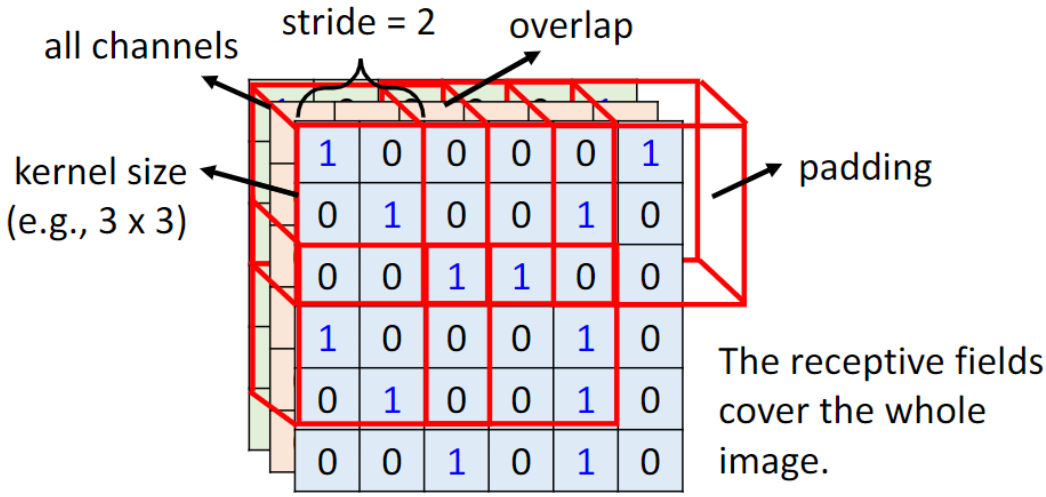


这里提出了**感受野（Reception field）**的概念，设定一个区域，只让一个特定的neuron关心这部分区域，把这里3×3×3的向量拉直，变成27维的列向量，再送入下一层神经元中。每一个neuron都有自己对应的Reception Field。另外，感受野之间范围亦可以重叠的；多个neuron也可以守备同个感受野；感受野的范围可以调整（依据pattern）以及感受野可以cover某些/一个channel（一般CNN不太常用）；区域也可以是长方形。



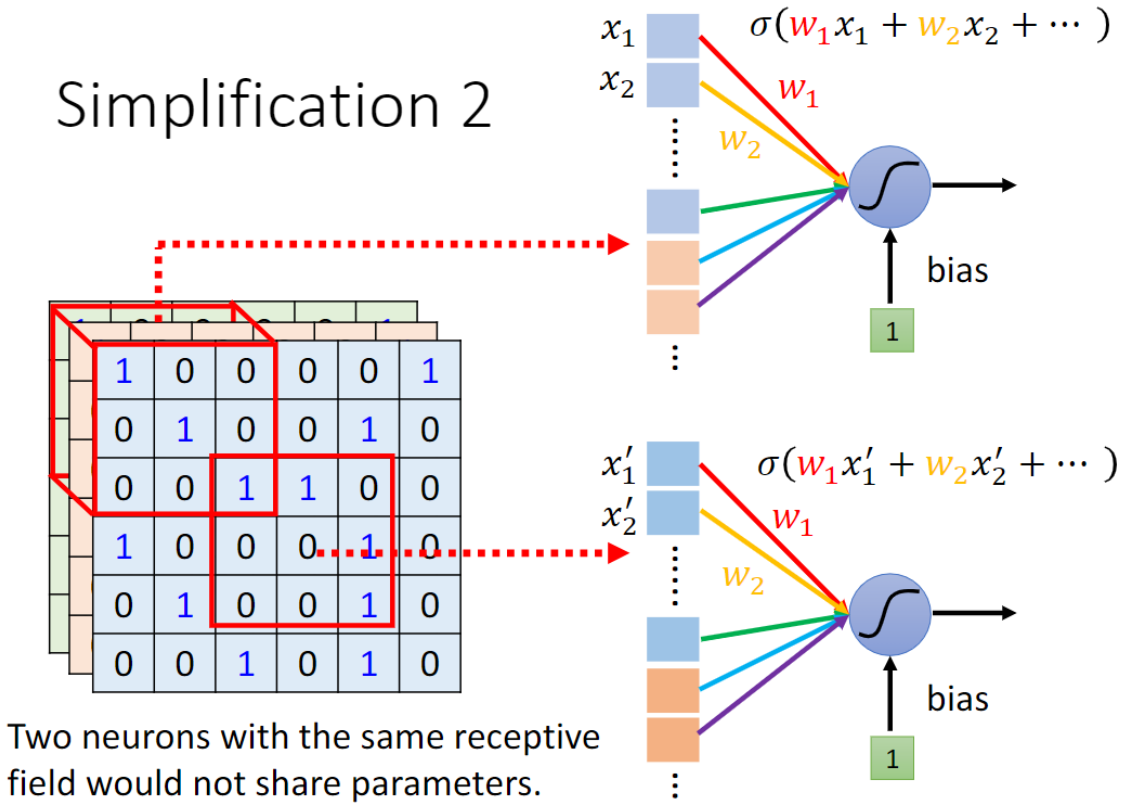
最经典的感受野设计方式：只考虑其**高和宽（kernel size）**，通常3×3.不需要考虑channel（即默认all channel）。一个Receptive Field都有一组神经元去守备。这个kernel改变的相对位置，其移动的量称之为**Stride**，一般Stride设计为1或2。在有序扫图片时有可能会超出本身图像尺寸，这里用到**padding**——超出部分补值（最普遍的是补0）。

Each receptive field has a set of neurons (e.g., 64 neurons).



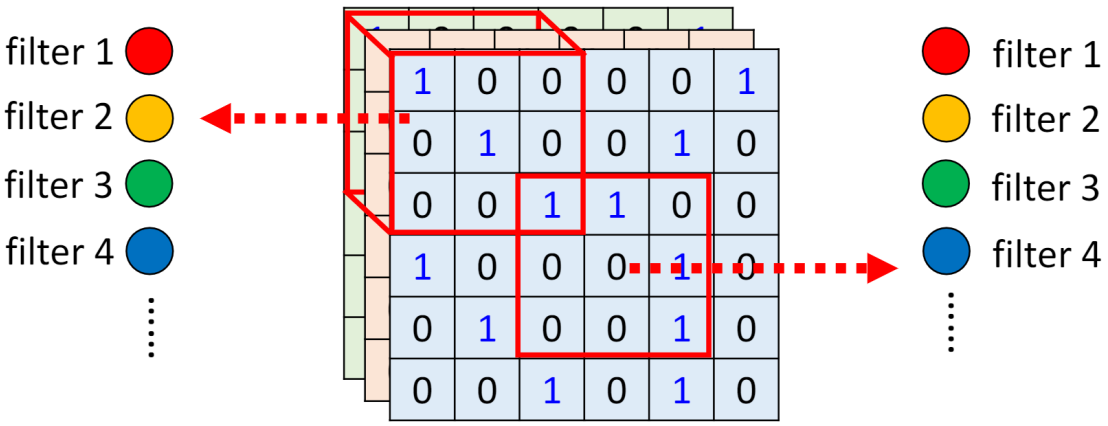
- *Observation 2*：The same pattern appear in different regions

  同样的pattern（局部特征）在图像中的坐标位置不同，那么我们能不能让不同的receptive field一起共享参数？（**Parameters sharing**）



上图中，相同颜色的权重weight是一样的，但是输出不会是一样的。让几个neuron一起 共享参数是这节的第二个简化。

典型的设计方式：1、每个receptive field都有一组neurons（比方说64个）；2、每隔receptive field都有神经元共享参数。

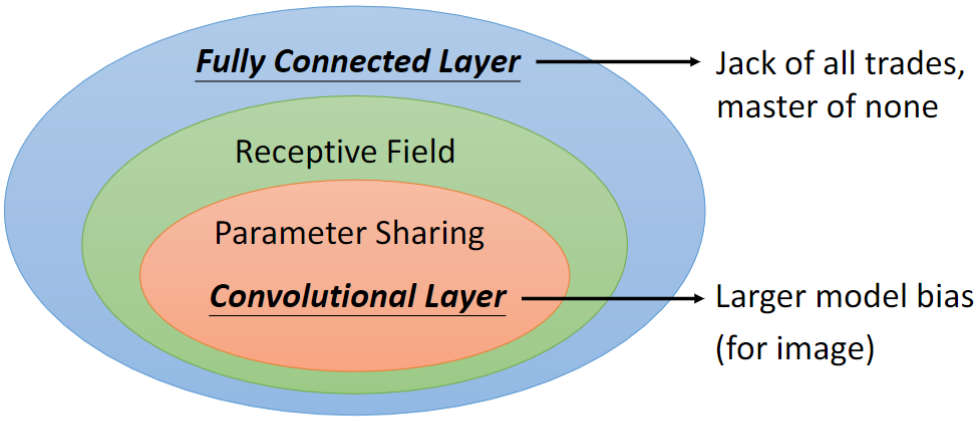


Benefit of Convolutional Layer

Fully Connected Layer全连接层（弹性最大）===>可以从一些patterns分类一张图像，得到receptive field（弹性变小） ==> 共享参数。至此，我们可以得到（定义了）：

Convolutional Layer = Receptive Field + Parameter Sharing

包含了**Convolutional Layer**的network就是**Convolutional Neuron Network（CNN）**。可以看到虽然CNN的bodel bias很大，但是由于CNN的设计特点是针对图像识别的，所以能在图像分类任务上表现不俗。



- Some patterns are much smaller than the whole image.
- The same patterns appear in different regions.

上述中所说的"共享参数"即filter（size:3×3×channel），neuron的权重weight就是张量filter的组成数值内容。Filter的高度就是待处理图像的channel，但第二层的filter不一样。考虑一定范围的filter真实考虑到的feature map的范围更大。**此处意义的卷积（Convolutional）：一个filter扫过整张图像。**（或者不同的neurons间可以共享参数）

- *Observation 3*：Subsampling the pixels will not change the object

  池化（Pooling）：图像下采样，把一张大的图像缩小。像一个激活函数，里面没有要learn的东西，是一个operator。最主要的理由是减少运算量，如果算力足够，pooling可有可无。

  - Pooling - Max Pooling

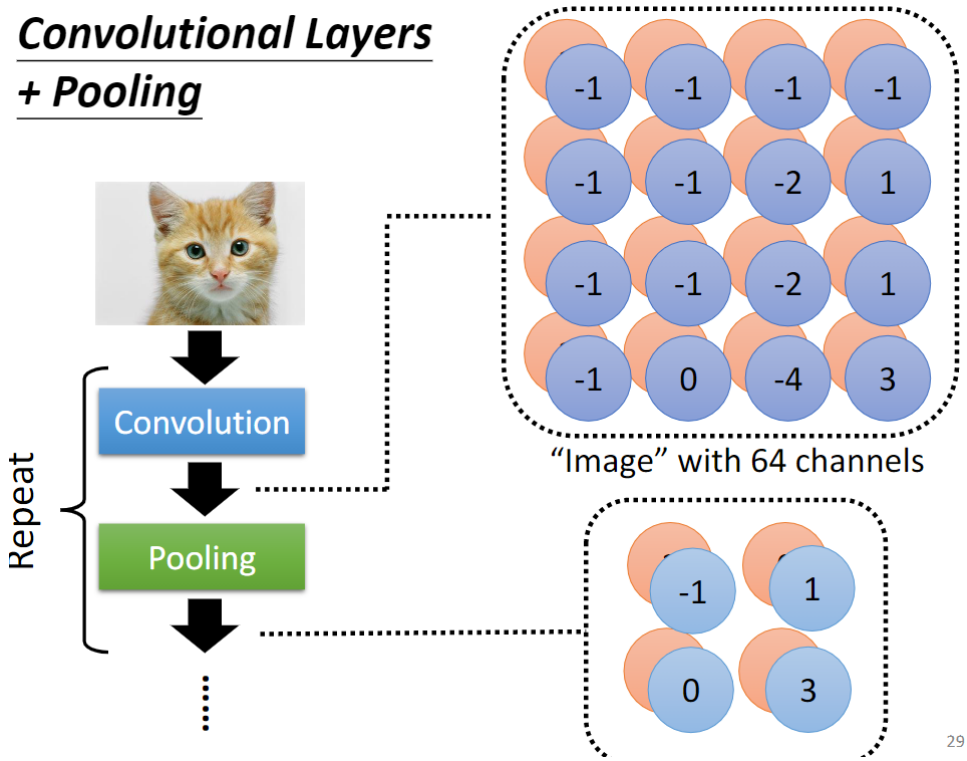
    分组：若干个一组，每组选最大的代表该组

  - Pooling - Mean Pooling

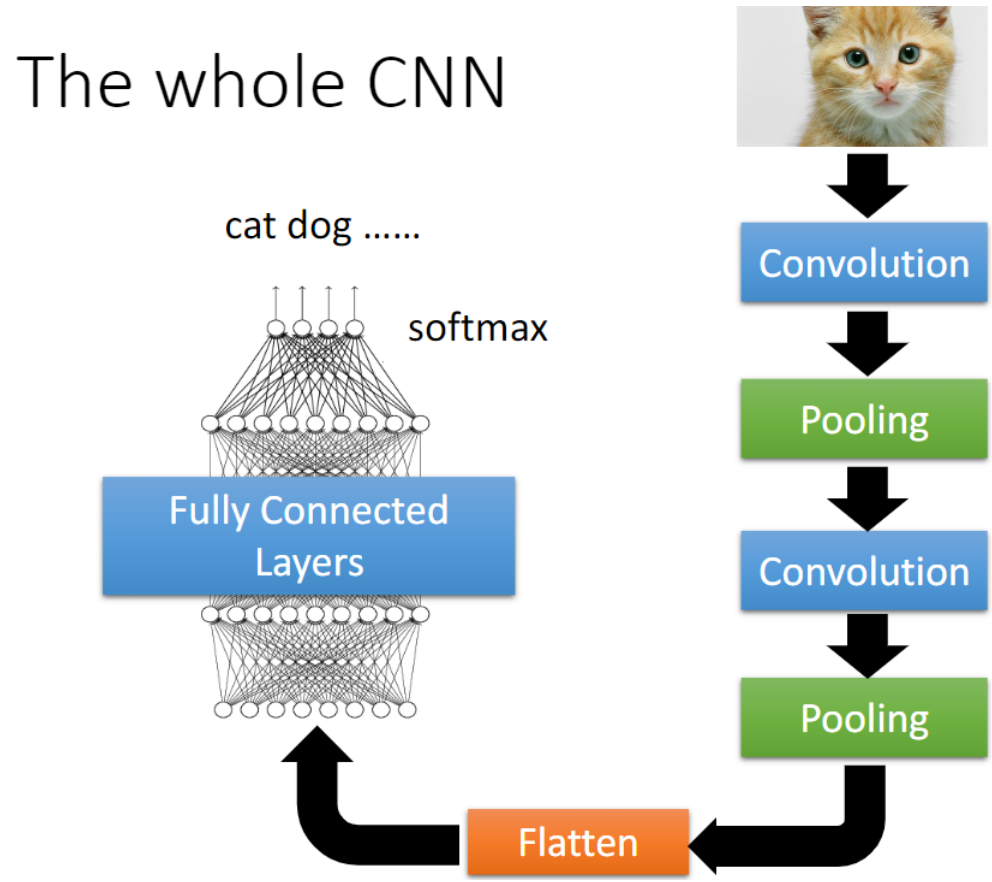
    分组：若干个一组，每组选平均值代表该组

  - 各式各样的pooling方法都有，如何分组有自己决定

  Convolutional Layer + Pooling：图像===>卷积层 ==>Pooling



下面展示整个CNN的样子：



flatten的作用就是把图像中的矩阵拉直变成一个向量，然后输入一个Fully Connected Layers，之后可能要经过一个$softmax$，然后最终得到一个图像辩识的结果。

***CNN用来下围棋**

把棋盘中的盘势当作向量（19×19 vector），落子（Black：1；white：-1；none：0）输入network，当作一个19×19的分类问题。

什么地方（符合什么样特征的任务）可以适合用CNN呢?

- Some pattern are much smaller than the whole image
- The same patterns appear in different regions
- Subsamplingthe pixels will not change the object（下围棋例如alphaGo不用pooling）

针对不同的任务，CNN中的具体设计肯定稍有不同

缺点

- CNN is not invariant to scaling and rotation (we need data augmentation ☺图像放大后适合原图的训练的CNN网络失效了。CNN不能够处理图像的放大缩小，或者旋转。因此，在图像识别中，往往要做Data Augmentation（前面讲过的，数据扩增）

  Spatial Transformer Layer可以解决这个问题。