在学习CNN之前，先学习DNN的知识

深度神经网络（DNN）是深度学习的基础，首先要理解DNN模型

DNN模型

感知机基本结构

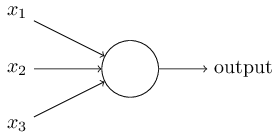


图1 感知机原理

输入输出学习到一个线性关系，得到中间输出结果，该模型只能用于二元分类，且无法学习较复杂的非线性模型。

神经网络在感知机上的拓展

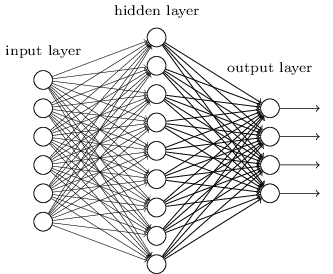


图2 神经网络

1. 加入了隐藏层（可以有多层），增强模型表达能力和模型的复杂度。
2. 输出层的神经元也变为多输出，这样模型可以灵活地应用于分类回归。
3. 对激活函数的拓展，感知机的激活函数是sign(z),在神经网络中一般用到Sigmoid函数，还有后来出现的tanx, softmax,和ReLU等。通过使用不同的激活函数，神经网络的表达能力进一步增强。

DNN基本结构

DNN内部的神经网络层可以分为三类，输入层，隐藏层和输出层,一般来说第一层是输入层，最后一层是输出层，而中间的层数都是隐藏层。层与层之间是全连接的，也就是说，第i层的任意一个神经元一定与第i+1层的任意一个神经元相连。虽然DNN看起来很复杂，但是从小的局部模型来说，还是和感知机一样。

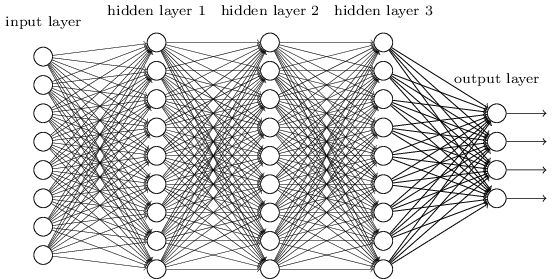


图3 DNN基本结构

前向传播算法

以三层DNN模型为例

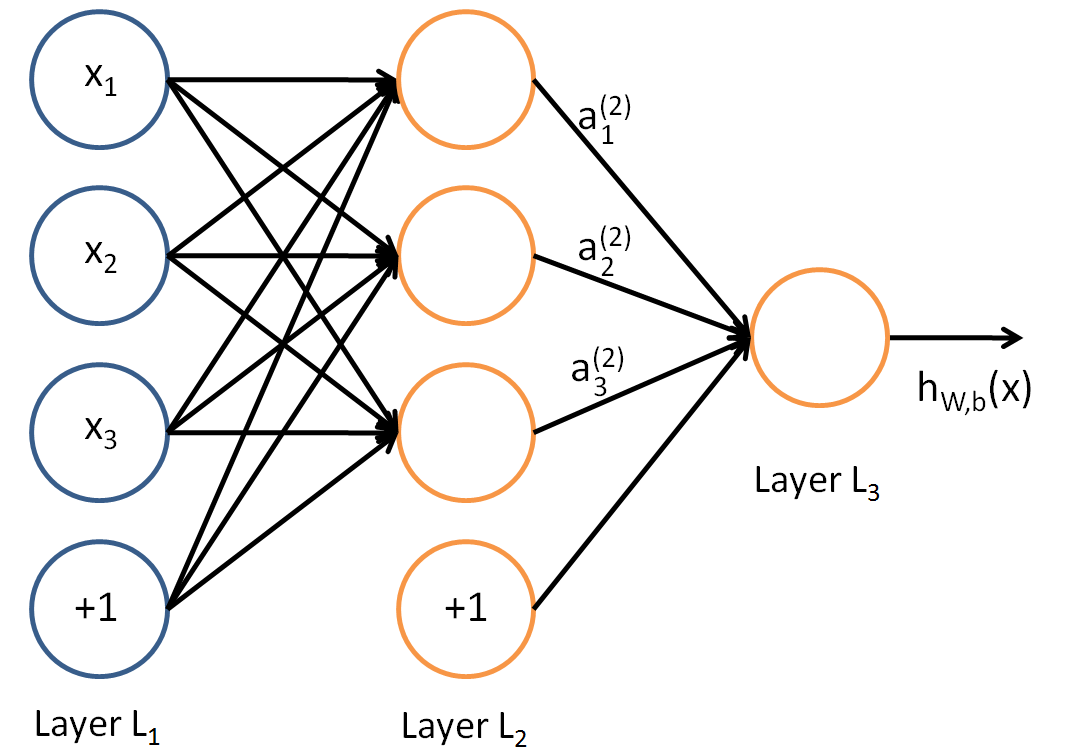


图4 算法原理

对于第二层的输出,,可以得到

对于第三层的输出可得，

将上面的例子一般化，假设第（l-1）层共有m个神经元，则对于第l层的第j个神经元的输出，可得：

用代数法计算较为复杂，用矩阵的方法表示如下,后续算法都基于矩阵法表示：

所谓的DNN的前向传播算法也就是利用若干个权重系数矩阵W,偏倚向量b来和输入值向量x进行一系列线性运算和激活运算，从输入层开始，一层层的向后计算，一直到运算到输出层，得到输出结果为值。

反向传播算法

DNN模型需要找到合适的所有隐藏层和输出层对应的线性系数矩阵W,偏倚向量b,让所有的训练样本输入计算出的输出尽可能的等于或很接近样本输出。算法优化过程一般用一个合适的损失函数来度量训练样本的输出损失，接着对这个损失函数进行优化求最小化的极值，对应的一系列线性系数矩阵WW,偏倚向量bb即为我们的最终结果。在DNN中，损失函数优化极值求解的过程最常见的一般是通过梯度下降法来一步步迭代完成的。

DNN损失函数和激活函数

（1）如果使用sigmoid激活函数，则交叉熵损失函数一般肯定比均方差损失函数好。

（2）如果是DNN用于分类，则一般在输出层使用softmax激活函数和对数似然损失函数。

（3）ReLU激活函数对梯度消失问题有一定程度的解决，尤其是在CNN模型中。

CNN模型

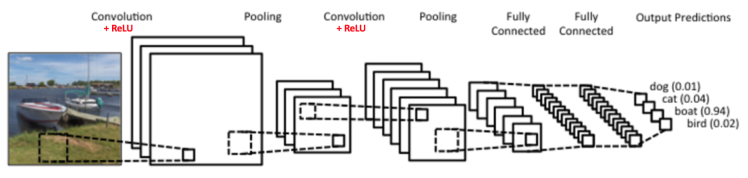


图5 CNN基本结构

图中是一个图形识别的CNN模型。可以看出最左边的图像就是输入层，计算机理解为输入若干个矩阵，这点和DNN基本相同。

　　接着是卷积层（Convolution Layer）,这个是CNN特有的，卷积层的激活函数使用的是ReLU。它其实很简单，就是ReLU(x)=max(0,x)。在卷积层后面是池化层(Pooling layer)，这个也是CNN特有的，需要注意的是，池化层没有激活函数。

　　卷积层+池化层的组合可以在隐藏层出现很多次，上图中出现两次。而实际上这个次数是根据模型的需要而来的。当然我们也可以灵活使用使用卷积层+卷积层，或者卷积层+卷积层+池化层的组合，这些在构建模型的时候没有限制。但是最常见的CNN都是若干卷积层+池化层的组合，如上图中的CNN结构。

　　在若干卷积层+池化层后面是全连接层（Fully Connected Layer, 简称FC），全连接层其实就是之前讲的DNN结构，只是输出层使用了Softmax激活函数来做图像识别的分类，这点在DNN中也有用到。

从上面CNN的模型描述可以看出，CNN相对于DNN，比较特殊的是卷积层和池化层，如果我们熟悉DNN，只要把卷积层和池化层的原理搞清楚了，那么搞清楚CNN就容易很多了。

卷积层

其实就是对输入的图像的不同局部的矩阵和卷积核矩阵各个位置的元素相乘，然后相加得到。基本算法如下图所示，

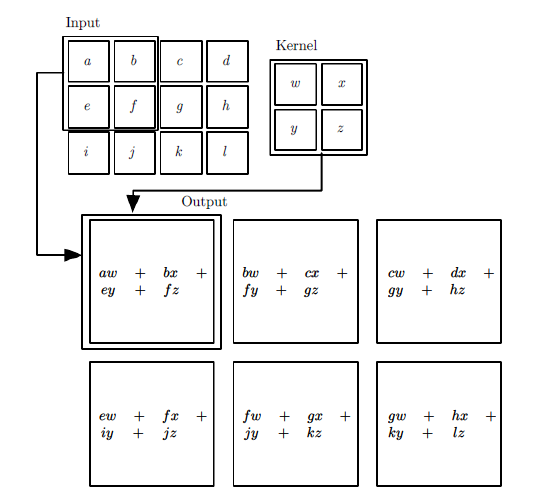


图6 卷积计算

把卷积过程用数学公式表达如下, 对于卷积后的输出，一般会通过ReLU激活函数，将输出的张量中的小于0的位置对应的元素值都变为0。

池化层

相比卷积层的复杂，池化层则要简单的多，所谓的池化，个人理解就是对输入张量的各个子矩阵进行压缩。假如是2x2的池化，那么就将子矩阵的每2x2个元素变成一个元素，如果是3x3的池化，那么就将子矩阵的每3x3个元素变成一个元素，这样输入矩阵的维度就变小了。要想将输入子矩阵的每n\*n个元素变成一个元素，那么需要一个池化标准。常见的池化标准有2个，MAX或者是Average。即取对应区域的最大值或者平均值作为池化后的元素值。

CNN的目标检测

因为应用种类多样，CNN以算法类别细分，CNN主要应用可分为图像分类(image classification)、目标检测(object detection)及语义分割(semantic segmentation)。

目标检测：一张图像内可有一或多个目标物，目标物也可以是属于不同类别。算法主要能达到两种目的：找到目标坐标及识别目标类别。简单来说，就是除了需要知道目标是什么，还需要知道它在哪个位置。  
目标检测应用非常普遍，包含文章开头提到的人脸识别相关技术结合应用，或是制造业方面的瑕疵检测，甚至医院用于X光、超音波进行特定身体部位的病况检测等。目标识别的基础可想象为在图像分类上增加标示位置的功能，故学习上也不离图像分类的基础。不过目标检测所标示的坐标通常为矩形或方形，仅知道目标所在位置，并无法针对目标的边缘进行描绘，所以常用见的应用通常会以「知道目标位置即可」作为目标。  
 最常见的算法为YOLO及R-CNN。其中YOLO因算法特性具有较快的识别速度，目前已来到v3版本。R-CNN针对目标位置搜寻及辨识算法和YOLO稍有不同，虽然速度稍较YOLO慢，但正确率稍高于YOLO。使用目标检测进行识别，通常输入为一张图像，而输出为一个或数个文字类别和一组或多组坐标。

SSD模型

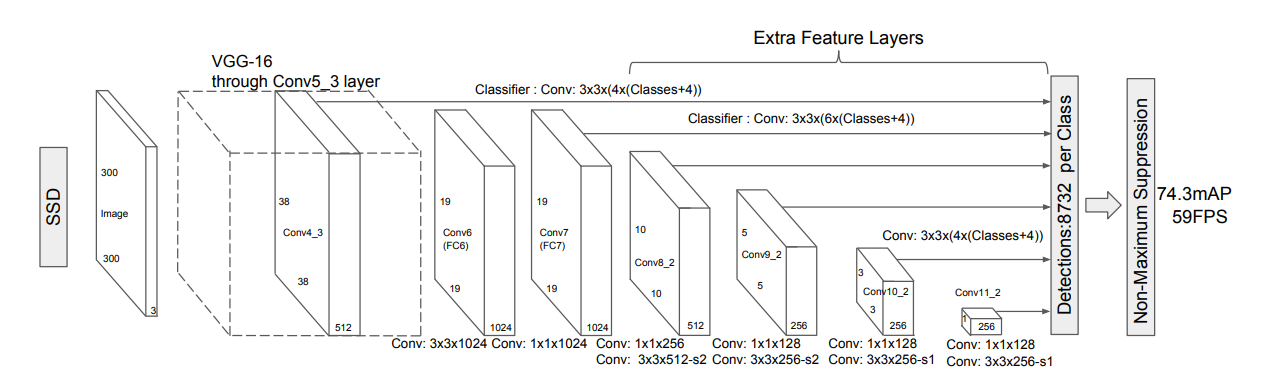
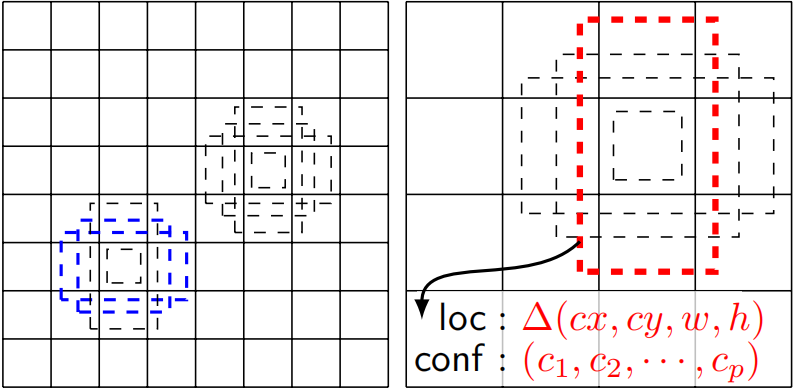
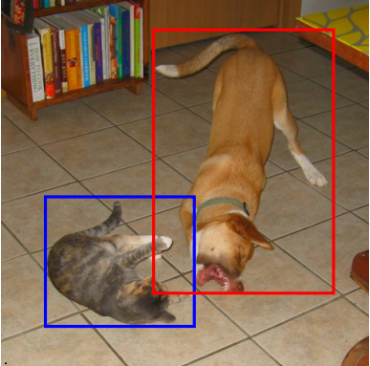


图7 SSD基本结构

模型说明：SSD采用VGG16作为基础模型，然后在VGG16的基础上新增了卷积层来获得更多的特征图以用于检测。图片通过vgg16的conv4\_3层得到一个38\*38的特征图1，对特征图1进行卷积，使用3\*3的卷积核，再使用1\*1的卷积核，即multi\_task方法，获得result\_sigmoid(h\*w\*1)和result\_softmax (h\*w\*m) 和result\_sigmoid对应的是这个区域的预选框是否包含正样本，result\_softmax对应的是这个区域那一个预选框为标准框的概率最大。以此类推，根据SSD自行设计的卷积操作得出之后的5个不同尺寸特征层。不同层特征层上，每个点上有预先设定数量，尺寸的目标框。将每个框的数据展平，通过全连接层进行目标类别判定。最后通过对比每个框目标准确率，矩形框重合率进行筛选最优的检测结果。

先验框



图中分为了很多网格，每一个网络都表达了一个感受野的概念，每一个网格都对应原图中的一块区域。一块区域就可以看做一组特征。然后对这些特征进行分类和回归。预选框是指在每一个像素点上都生成一系列固定大小但宽高比不同的框。若一张特征图有k个默认框，对每个默认框都需预测c种类别的分数和每个边框4条边的偏移值，若特征图大小为m\*n，则一张特征图的输出有(c+4)\*k\*m\*n个。

训练中一张完整的图片送进网络获得各个特征图，对于正样本训练来说，需要先将先验框与真实框做匹配，匹配成功说明这个先验框所包含的是个目标，但离完整目标的真实框还有段距离，训练的目的是保证预选框的分类的置信度的同时将先验框尽可能回归到真实框。

正负样本选择

正样本：在训练时，我们需要建立真实标签和默认框之间的对应关系。请注意，对于每个真实标签框，我们从默认框中进行选择，这些默认框随位置、纵横比和比例而变化。启始时，我们匹配每个真实标签框与默认框最好的jaccard 重叠。这是原始MultiBox使用的匹配方法，它确保每个真实标签框有一个匹配的默认框。与MultiBox不同，匹配默认框与真实标签jaccard重叠高于阈值（0.5）的默认框。添加这些匹配简化了学习问题：它使得有多个重叠默认框时网络预测获得高置信度，而不是要求它选择具有最大重叠的那个。

负样本：在匹配步骤之后，大多数默认框都是负样本，特别是当可能的默认框数量很大时。这导致了训练期间正负样本的严重不平衡。我们使用每个默认框的最高置信度对它们进行排序，并选择前面的那些，使得正负样本之间的比率最多为1：3，以代替使用所有的负样本。我们发现，这促使更快的优化和更稳定的训练。

损失函数

损失函数由分类和回归两部分组成，回归部分的loss是希望预测的box和先验框的差距尽可能跟真实框和先验框的差距接近，这样预测框就能尽量和真实框一样。

