



城市空间建模与仿真

第十一讲 城市空间三维数据表达和重建-几何体识别与体素神经网络

任课教师：汤圣君
建筑与城市规划学院 城市空间信息工程系

目录

CONTENTS

01

基于剪影轮廓的几何体识别

02

3D体素神经网络

目录

CONTENTS

01

基于剪影轮廓的几何体识别

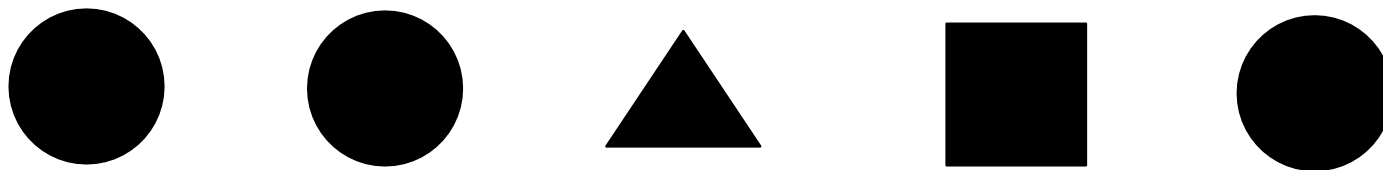
基于“剪影”轮廓的几何体识别

通过不同视角的剪影识别几何体

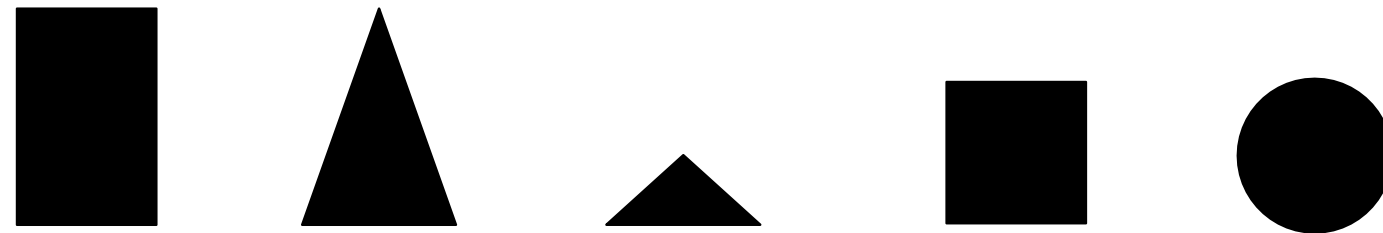


- 为什么不直接在3D数据上识别?
- 因为运算简单
 - 当我们有了3D数据后, 就能够方便地生成各个角度的投影
 - 不同的几何体, 在不同方向的投影有一定特点, 可以用于识别

俯视图



正视图



能够识别出以上投影对应的他们的3D几何体吗?

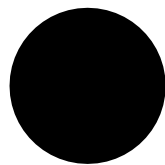
基于“剪影”轮廓的几何体识别

通过不同视角的剪影识别

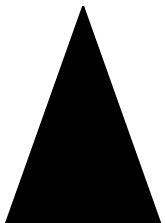
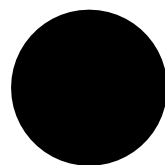


俯视图

正视图



圆柱体



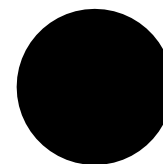
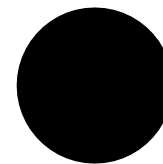
圆锥体



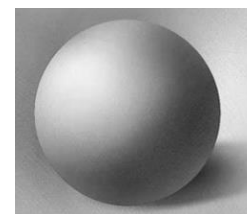
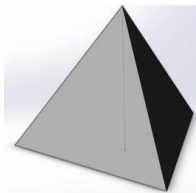
四面体



立方体



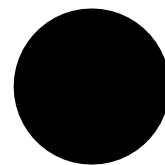
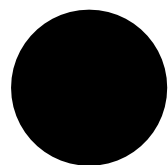
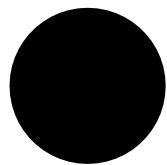
球



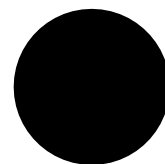
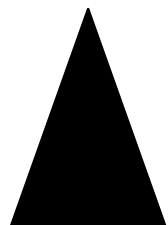
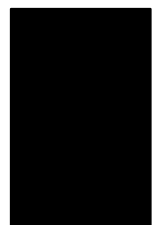
基于“剪影”轮廓的几何体识别

通过不同视角的剪影识别

俯视图



正视图



圆柱体

圆锥体

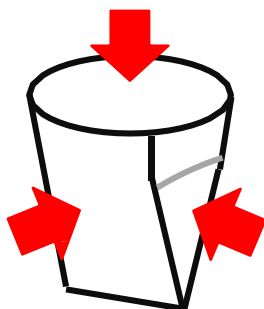
四面体

立方体

球



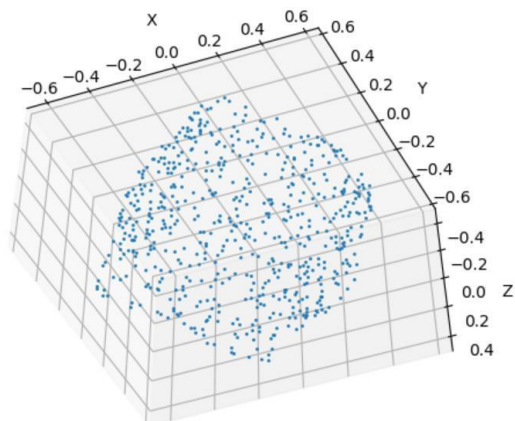
对形状有
多大把握？



视角越多，分类的把握越大

基于“剪影”轮廓的几何体识别

通过不同视角的剪影识别



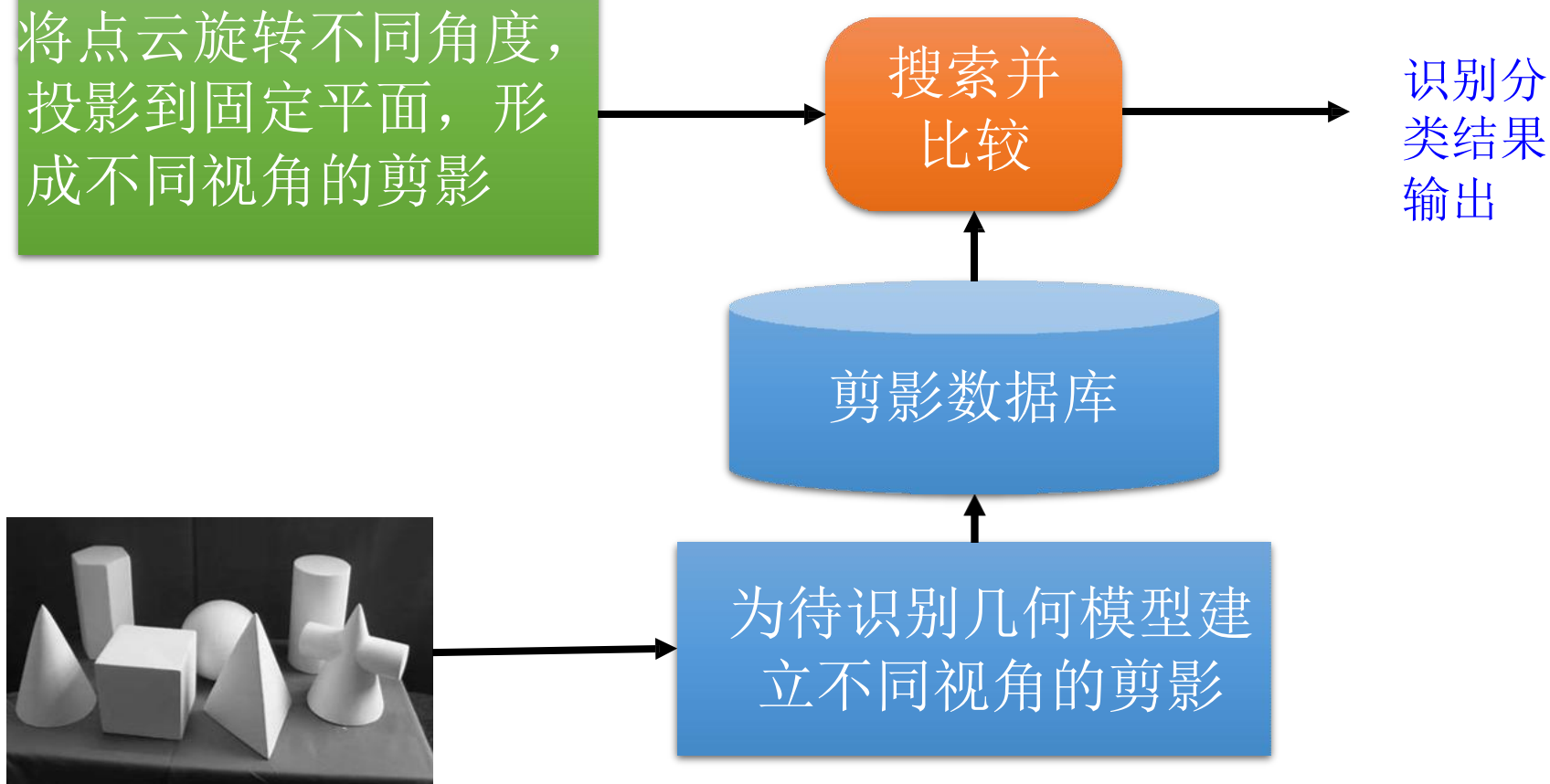
将点云旋转不同角度，投影到固定平面，形成不同视角的剪影



• 如何搜索？



• 后面通过手势识别的简单例子介绍整个步骤，以及其中的算法优化



待识别分类的几何体模型数据

基于“剪影”轮廓的几何体识别

应用实例——静态手势识别

- 识别手部动作



基于“剪影”轮廓的几何体识别

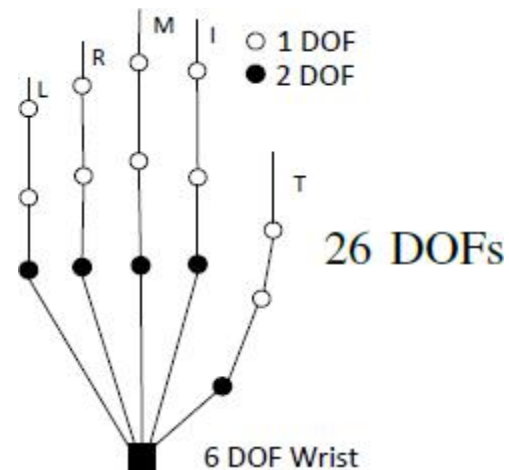
应用实例——静态手势识别技术途径

• 基于骨架模型

- 分两步进行：
 - 1 识别手部关节骨架模型参数
 - 2 识别手部动作分类
- 能够识别精细动作
- 效率低，运算量大，受自遮挡影响

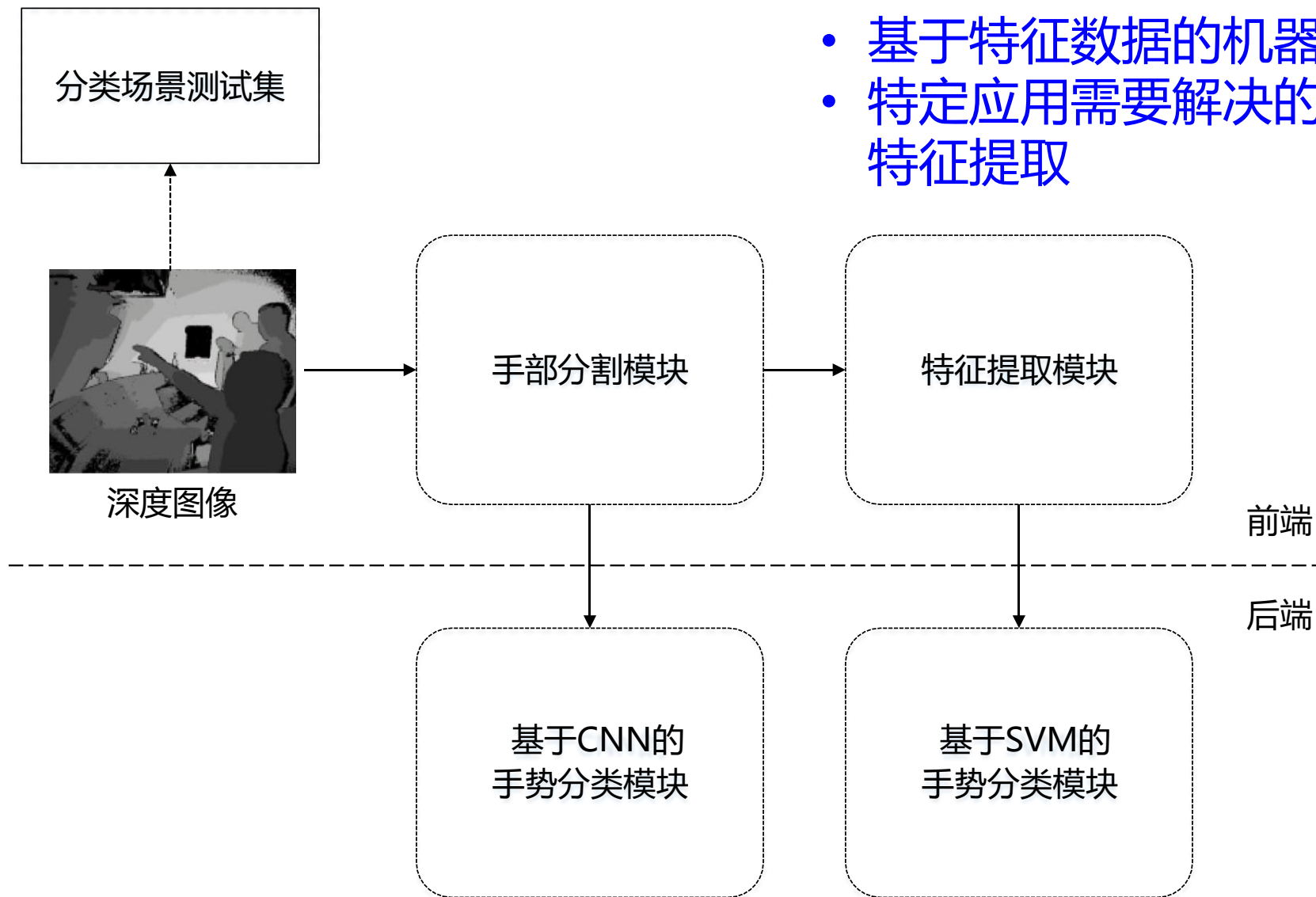
• 基于剪影轮廓

- 分两步
 - 手部分割，获取剪影
 - 剪影形状分类
- 识别明显的手势动作
- 效率高，运算量小
- 受自遮挡影响，精细动作区分困难



基于“剪影”轮廓的几何体识别

应用实例——静态手势识别算法流程

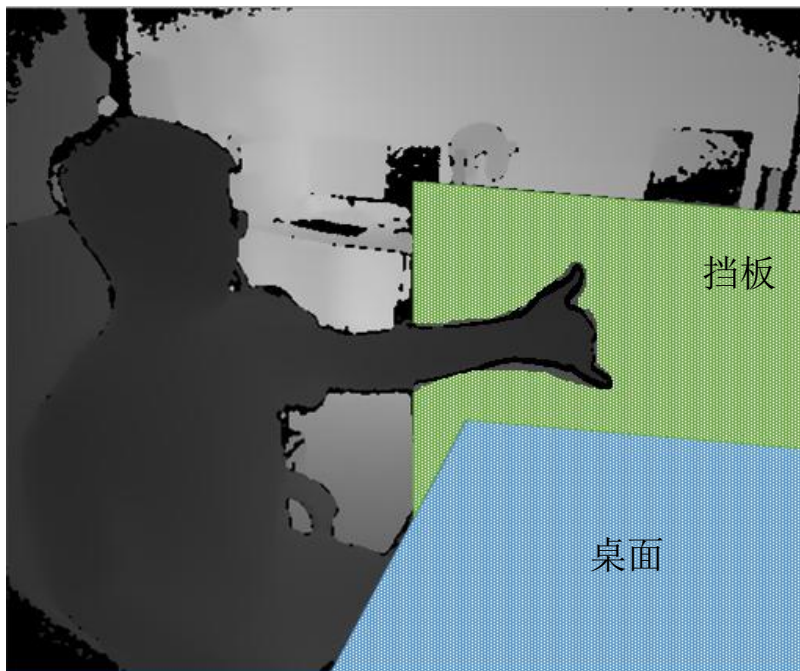


- 基于特征数据的机器学习算法相对成熟
- 特定应用需要解决的困难是目标分割和特征提取

基于“剪影”轮廓的几何体识别

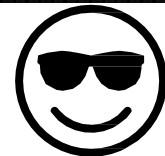
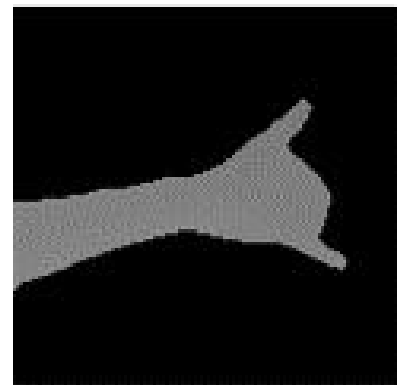
静态手势识别——前景提取

- 从深度图中提取手部像素

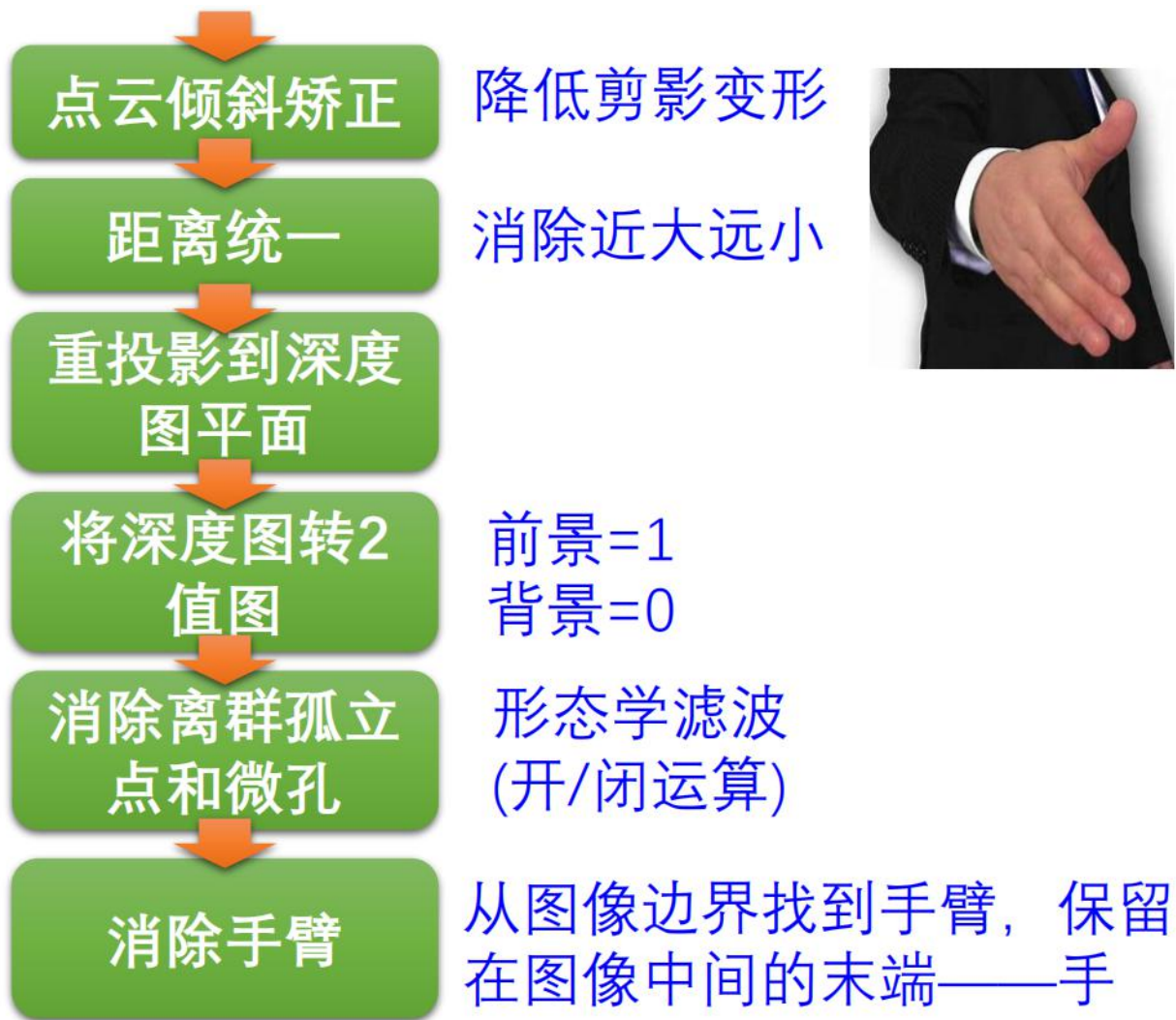


采取多种措施滤除背景

- 😊 • 基于距离的点云过滤
近距离的桌面怎么办? 😞
- 😊 • 基于背景扣除的点云过滤
背景的噪声怎么办? 😞
- 😊 • 基于统计建模的背景去除
残留的噪点怎么办? 😞
- 😊 • 深度图上离群点过滤(形态学滤波)



基于“剪影”轮廓的几何体识别



深度图→点云→旋转→深度图



- 如何获得旋转量?
- 把连着手臂的点云看成椭球体的话，希望他的长轴旋转到平行于相机传感器平面



基于“剪影”轮廓的几何体识别



- ☹️ • 如何将剪影变成可以比较的数字?
- 😊 • Hu矩——将剪影转成7维向量
 - 旋转不变
 - 平移不变
 - 缩放不变

基于“剪影”轮廓的几何体识别

图像的Hu不变矩特征

下面依次给出几个数学概念和运算方法

- 矩(Moment): 概率与统计中的概念, 是随机变量的一种数字特征

$$m_{pq} = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} x^p y^q f(x, y) \quad p, q = 0, 1, 2 \dots$$

$f(x, y)$ 是图像在位置 (x, y) 的(灰度)值
 (N, M) 是图像尺寸
 对二值图 $f(x, y) = 0$ 或 1

构造7个不变矩,
 具有平移、旋转和尺度不变特性

- 图像中心(重心)

$$\bar{x} = m_{10}/m_{00}$$

$$\bar{y} = m_{01}/m_{00}$$

把 $f(x, y)$ 看作像素重量的话, 它给出了图像“重心”

- 中心矩

$$\mu_{pq} = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} (x - \bar{x})^p (y - \bar{y})^q f(x, y)$$

$$p, q = 0, 1, 2 \dots$$

- 归一化中心矩

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma}$$

$$\gamma = \frac{p+q}{2} + 1, \quad p+q = 2, 3, \dots$$

$$h_0 = \eta_{20} + \eta_{02}$$

$$h_1 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$h_2 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$h_3 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$h_4 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\ + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)$$

$$h_5 = (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\ + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$h_6 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\ - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)$$

基于“剪影”轮廓的几何体识别

图像剪影的Hu不变矩特征

7个Hu不变矩

$$\begin{aligned}
 h_0 &= \eta_{20} + \eta_{02} \\
 h_1 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
 h_2 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
 h_3 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
 h_4 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\
 &\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\
 h_5 &= (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) \\
 &\quad + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
 h_6 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) \\
 &\quad - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2)
 \end{aligned}$$



对数变换将其转到接近的数量级

$$H_i = -\text{sign}(h_i) \log |h_i|$$

$$h_0 = 0.00162663$$

$$h_1 = 3.11619 \times 10^{-07}$$

$$h_2 = 3.61005 \times 10^{-10}$$

$$h_3 = 1.44485 \times 10^{-10}$$

$$h_4 = -2.55279 \times 10^{-20}$$

$$h_5 = -7.57625 \times 10^{-14}$$

$$h_6 = 2.09098 \times 10^{-20}$$

数量级差别太大，不利于特征比较

$$H_0 = 2.78871$$

$$H_1 = 6.50638$$

$$H_2 = 9.44249$$

$$H_3 = 9.84018$$

$$H_4 = -19.593$$

$$H_5 = -13.1205$$

$$H_6 = 19.6797$$

代码实现：







```

11 m = cv2.moments(img)
12 h = cv2.HuMoments(m)
13 H = -np.sign(h) * np.log10(np.abs(h))

```

基于“剪影”轮廓的几何体识别

图像剪影的Hu不变矩特征

id	Image	H[0]	H[1]	H[2]	H[3]	H[4]	H[5]	H[6]
K0		2.78871	6.50638	9.44249	9.84018	-19.593	-13.1205	19.6797
S0		2.67431	5.77446	9.90311	11.0016	-21.4722	-14.1102	22.0012
S1		2.67431	5.77446	9.90311	11.0016	-21.4722	-14.1102	22.0012
S2		2.65884	5.7358	9.66822	10.7427	-20.9914	-13.8694	21.3202
S3		2.66083	5.745	9.80616	10.8859	-21.2468	-13.9653	21.8214
S4		2.66083	5.745	9.80616	10.8859	-21.2468	-13.9653	-21.8214

- 不同的S图像有相同的H值
- 具有平移、旋转和尺度不变特性

<https://www.learnopencv.com/shape-matching-using-hu-moments-c-python/>

基于“剪影”轮廓的几何体识别

图像剪影的Hu不变矩特征



- 为标准手势（包括不同的变形）建立Hu不变矩模板（训练集），作为样本库保存
- 对输入手势经过前处理，形成尺寸归一化的剪影
- 计算其Hu不变矩
- 和样本库的Hu不变矩比较，找到最接近的输出

比较两个不变矩 $H_i^{(a)}$ 和 $H_i^{(b)}$ 的
多种方式(不止列出的这几种)

$$D(a, b) = \sum_{i=0}^6 |H_i^{(a)} - H_i^{(b)}|$$

$$D(a, b) = \sum_{i=0}^6 \left| \frac{1}{H_i^{(a)}} - \frac{1}{H_i^{(b)}} \right|$$

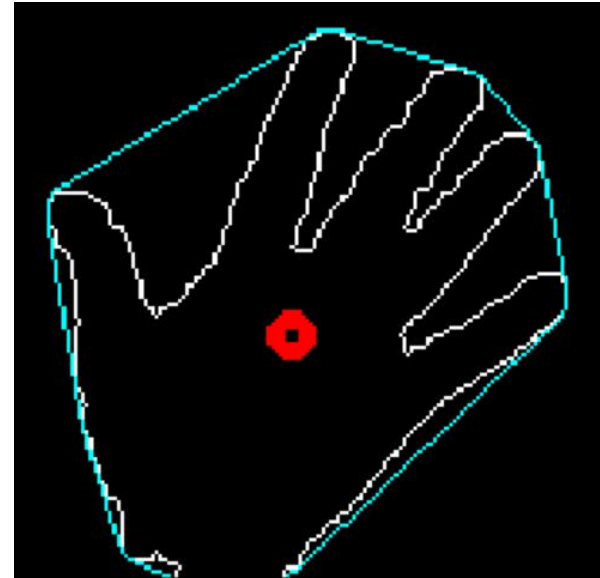
$$D(a, b) = \sum_{i=0}^6 \frac{|H_i^{(a)} - H_i^{(b)}|}{|H_i^{(a)}|}$$

$$D(a, b) = \sqrt{\sum_{i=0}^6 |H_i^{(a)} - H_i^{(b)}|^2}$$

基于“剪影”轮廓的几何体识别

图像剪影的Hu不变矩特征

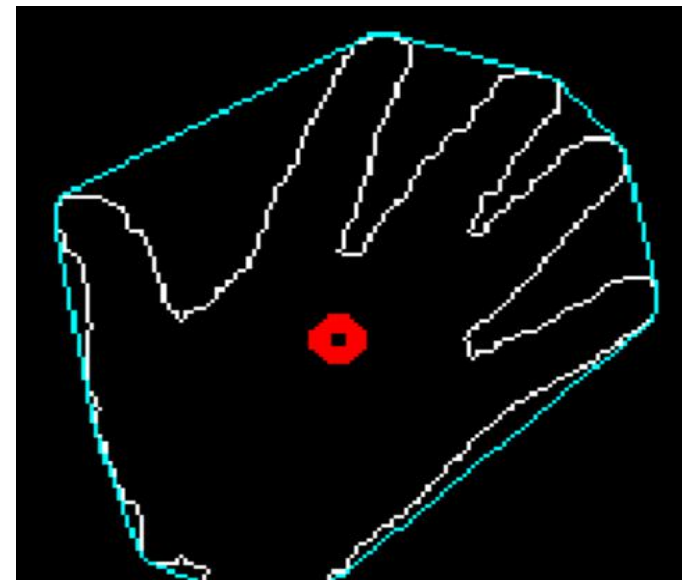
- ☹️ • 手形的变化、手指夹角改变、手掌切割误差、手指弯曲和手掌倾斜导致Hu不变矩的改变
- 😊 • 根据实际数据的算法调整
 - 使用剪影的轮廓替代剪影计算Hu不变矩
 - 使用剪影的凸多边形轮廓替代剪影计算Hu不变矩
- 近邻检索的更新
 - 替换成SVM分类器
 - 替换成神经网络分类器



基于“剪影”轮廓的几何体识别

静态手势识别——代码提示

- 深度图二值化: `cv2.threshold()`
- 找到外轮廓凸包: `cv2.convexHull()`
- 从二值图查找轮廓: `cv2.findContours()`
- 数值形态学滤波: `cv2.morphologyEx()`
- 形状比较: `cv2.matchShapes ()`



目录

CONTENTS

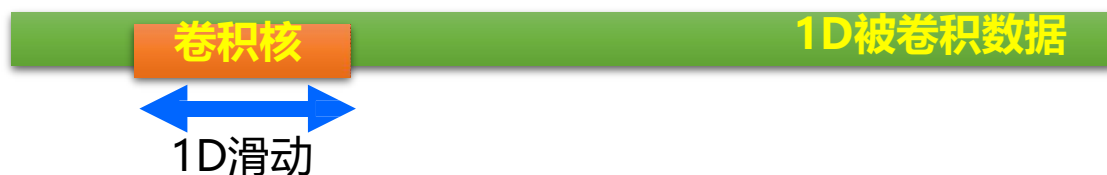
02

3D体素神经网络

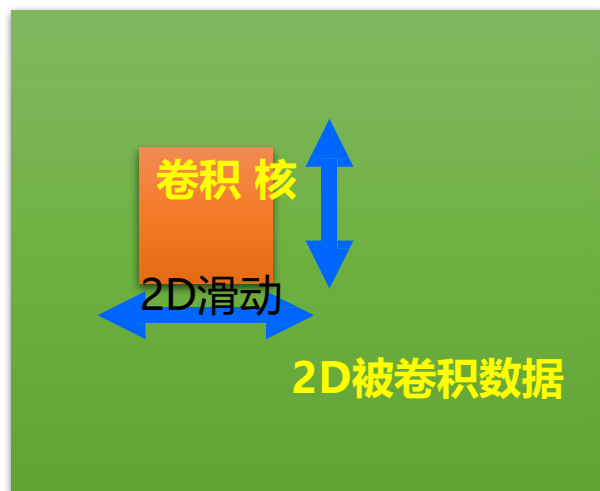
3D体素神经网络-前言

- 传统图像卷积神经网络是2D方向的卷积
- 现有神经网络框架(tensorflow)支持直接进行真正的3D卷积
- 下面是1D、2D和3D卷积的比较

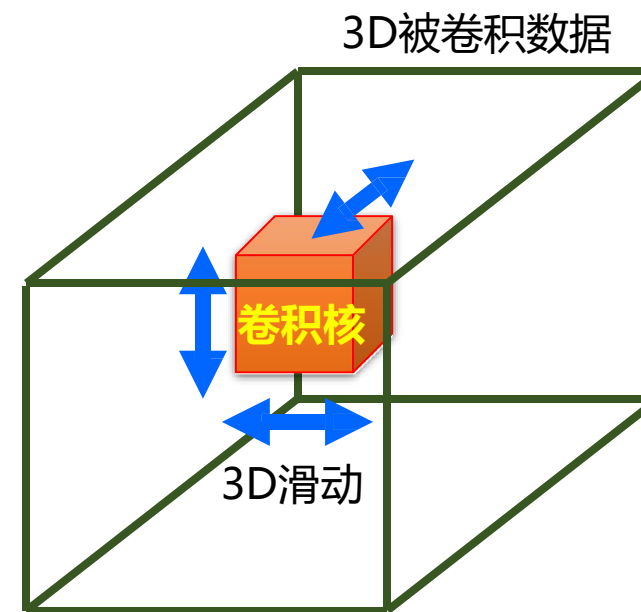
1D卷积



2D卷积



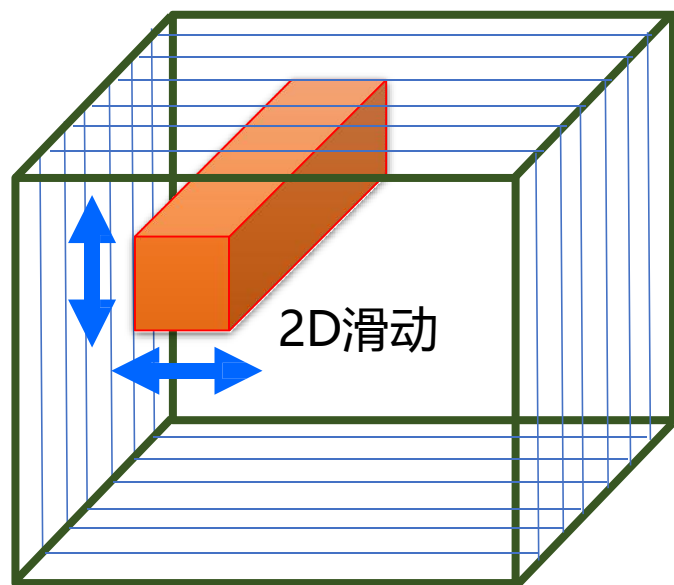
3D卷积



3D体素神经网络-前言

- 图像卷积神经网络中经常遇到“多通道”2D卷积，容易和3D卷积混淆，下面是两者的比较

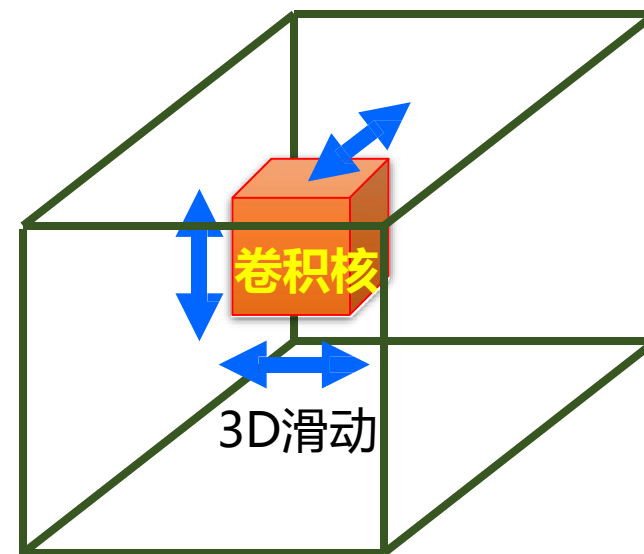
多通道2D卷积和3D卷积



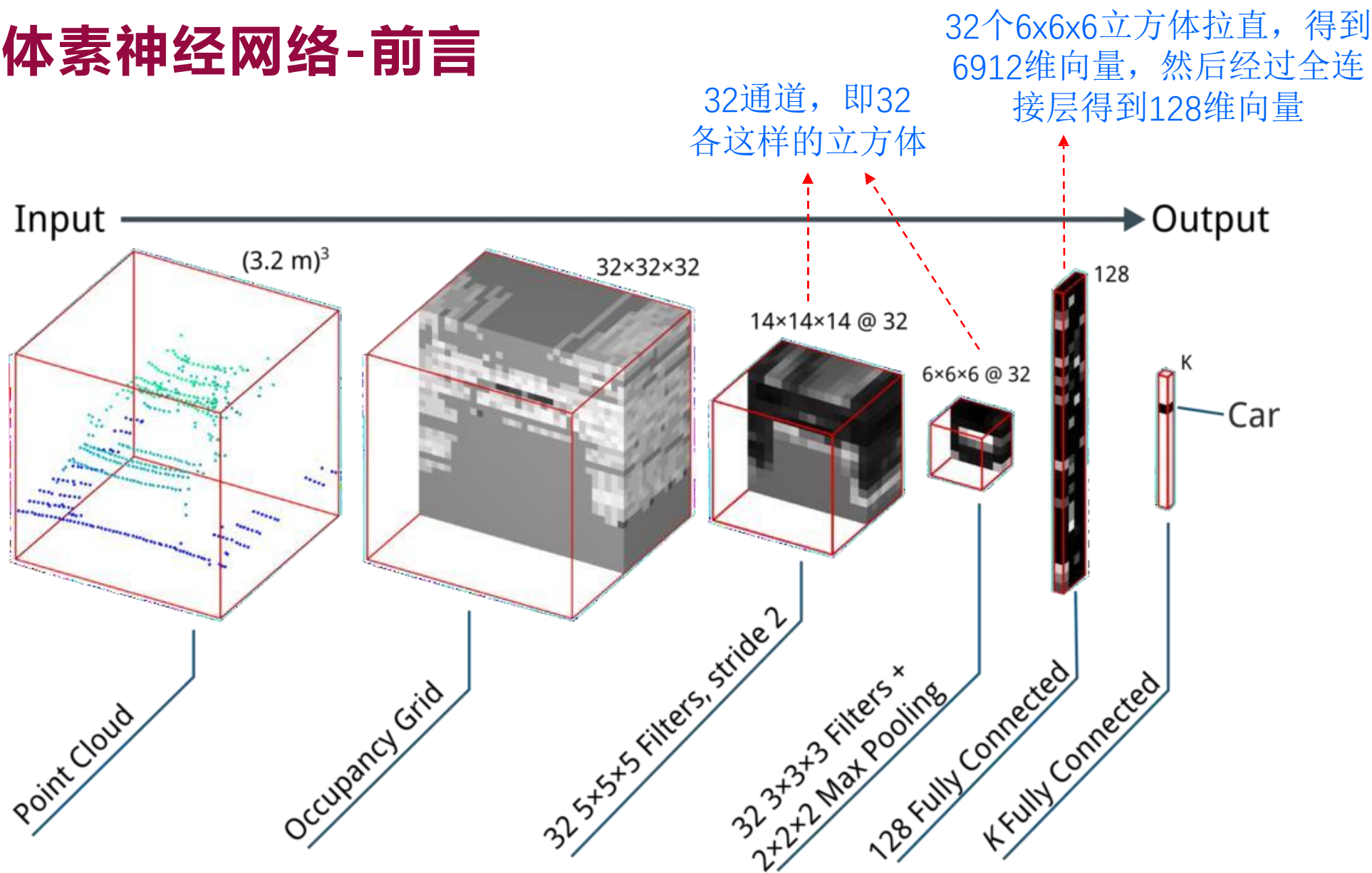
每个通道
是一个
2D图

3D卷积

3D被卷积数据

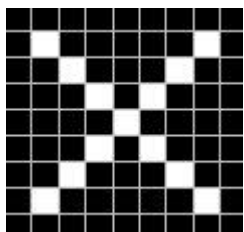


3D体素神经网络-前言

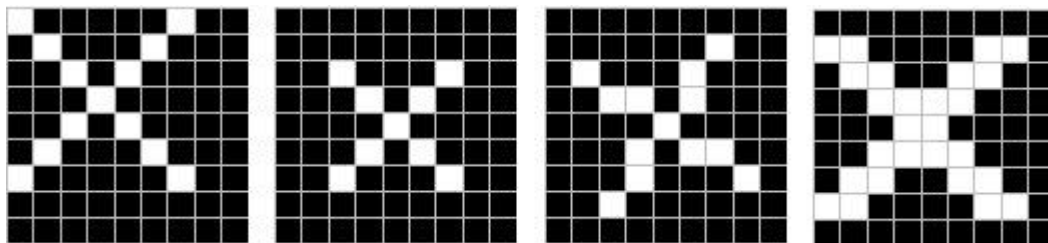


3D体素神经网络-卷积操作

- 我现在要训练一个最简单的CNN，用来识别一张图片里的字母是X还是O



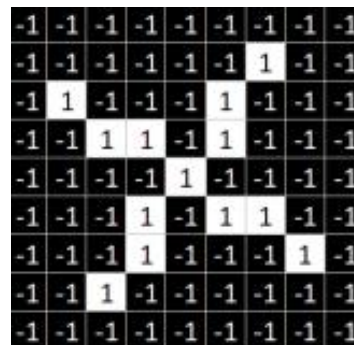
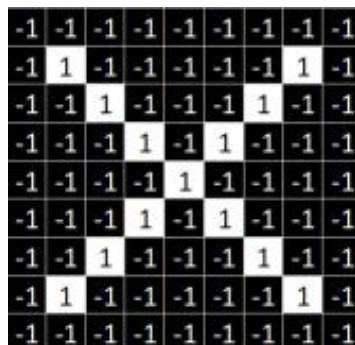
- 四个都是X，但它们和之前那张X明显不一样，计算机没见过它们，又都不认识了



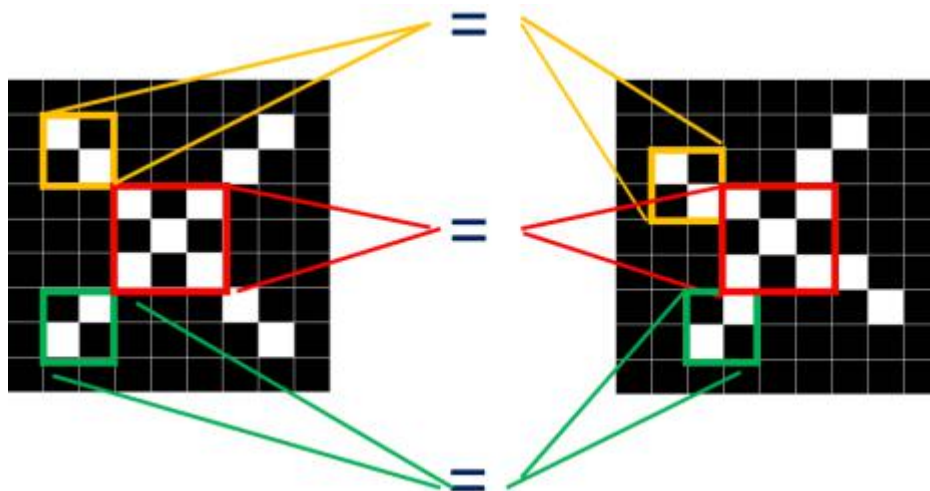
CNN要做的，就是如何提取内容为X的图片的特征

3D体素神经网络-卷积操作

- 如果按照每像素逐个比较肯定是不科学的，结果不对而且效率低下，因此提出其他匹配方法，称之为 patch 匹配



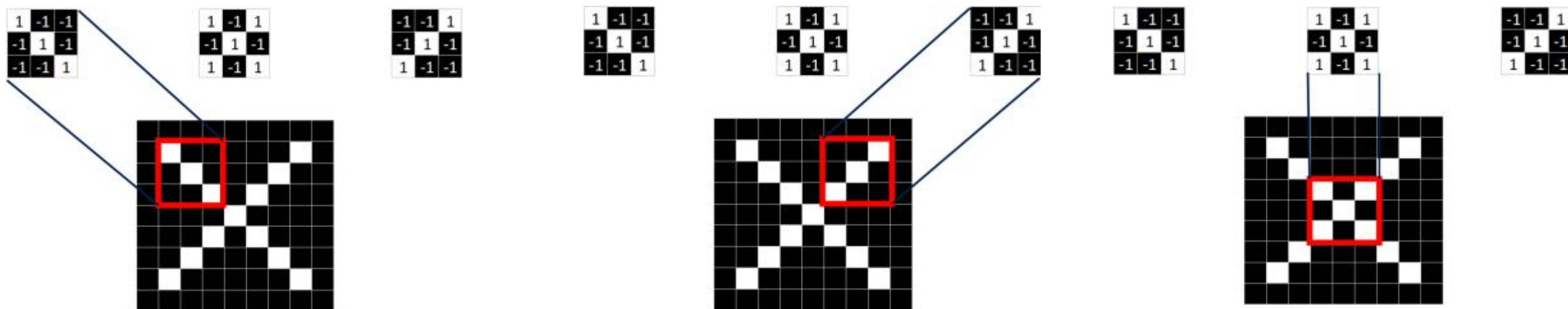
相当于如果我要在一张照片中
进行人脸定位，但是CNN不
知道什么是人脸，我就告诉它：
人脸上有三个特征，眼睛鼻子
嘴巴是什么样



- 如上图所示，两张图中三个同色区域的结构完全一致！
- 因此，我们就考虑，要将这两张图联系起来，无法进行全体像素对应，但是否能进行局部地匹配？

3D体素神经网络-卷积操作

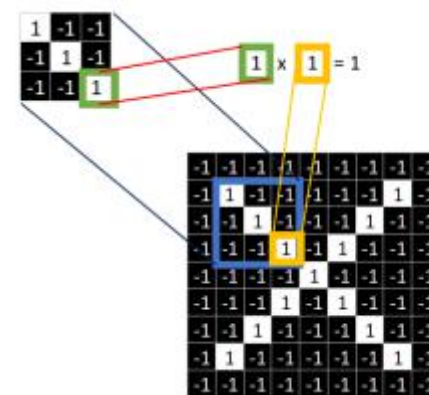
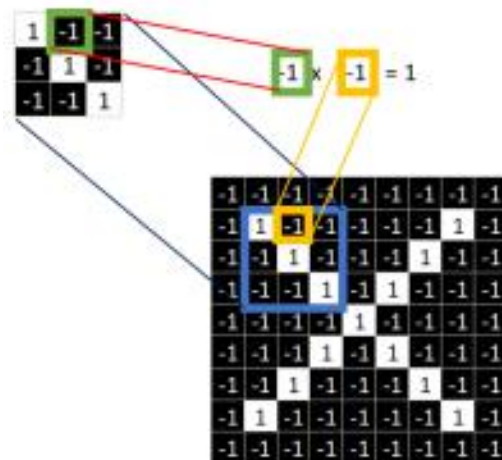
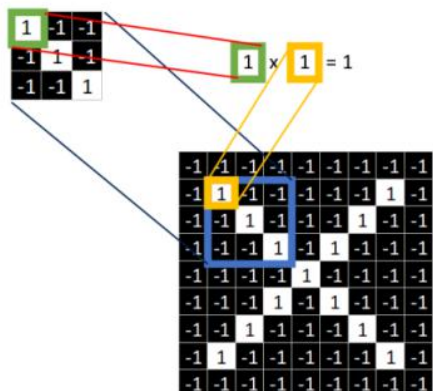
- 从标准的X图中我们提取出三个特征 (feature)



- 我们发现只要用这三个feature便可定位到X的某个局部, feature在CNN中也被成为卷积核 (filter), 一般是3X3, 或者5X5的大小

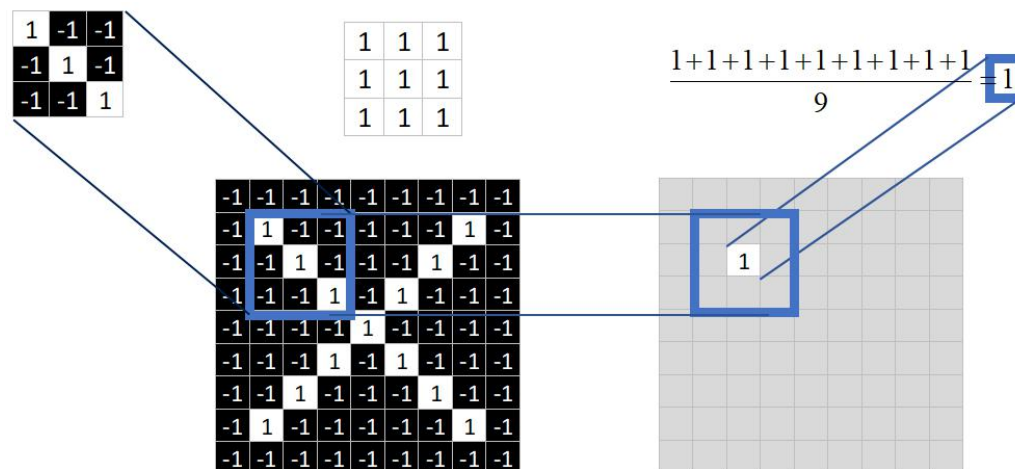
3D体素神经网络-卷积操作

- 卷积运算：卷积神经网络在本质和原理上还是和卷积运算有一定的联系的

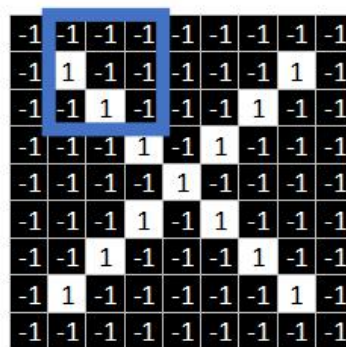
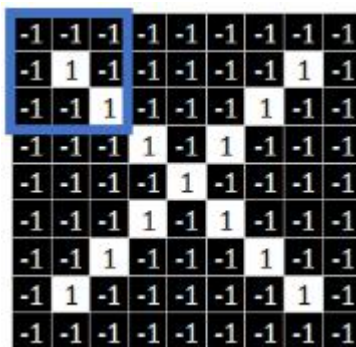


3D体素神经网络-卷积操作

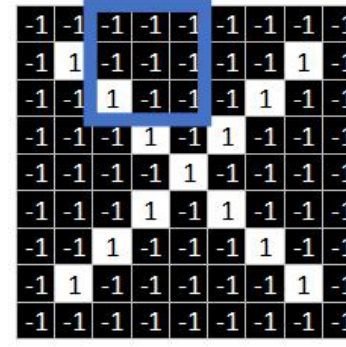
- 接下来的工作是对右图九个值求平均，得到一个均值，将均值填入一张新的图中。



- 卷积对应相乘运算并求得均值后，滑动窗便开始向右边滑动。根据步长的不同选择滑动幅度
 - 若 步长 $\text{stride}=1$ ，就往右平移一个像素
 - 若 步长 $\text{stride}=2$ ，就往右平移两个像素。



$\text{stride}=1$



$\text{stride}=2$

3D体素神经网络-卷积操作

- 经过一系列卷积对应相乘，求均值运算后，我们终于把一张完整的feature map填满

1	-1	-1
-1	1	-1
-1	-1	1

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

对图像运用该卷积核，
产生的结果

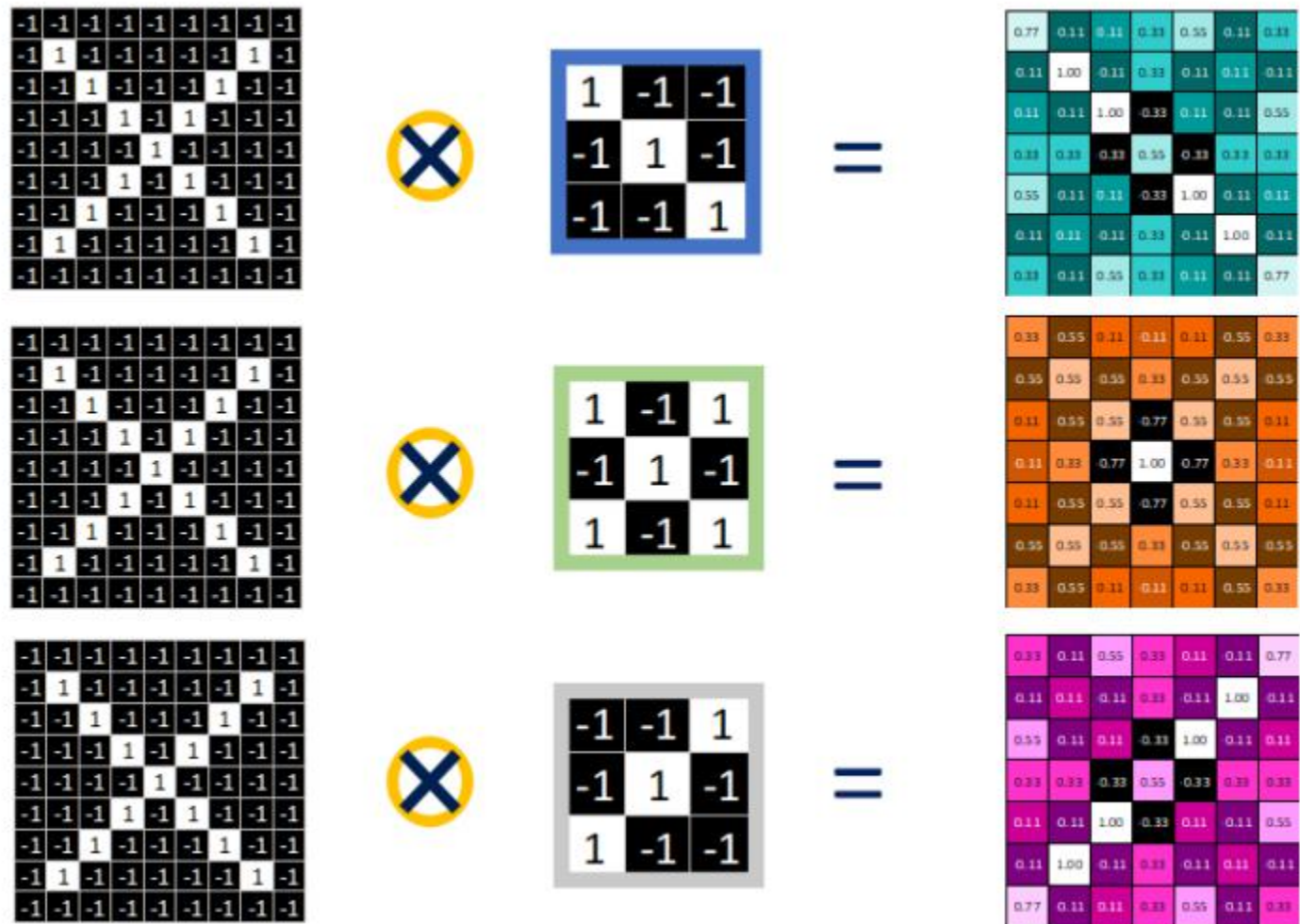


0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

- **feature map**是每一个feature从原始图像中提取出来的“特征”。其中的值，越接近为1表示对应位置和feature的匹配越完整，越是接近-1，表示对应位置和feature的反面匹配越完整，而值接近0的表示对应位置没有任何匹配或者说没有什么关联

3D体素神经网络-卷积操作

- 一个feature作用于图片产生一张feature map, 对这张X图来说, 我们用的是3个 feature, 因此最终产生3个 feature map,到此卷积结束。

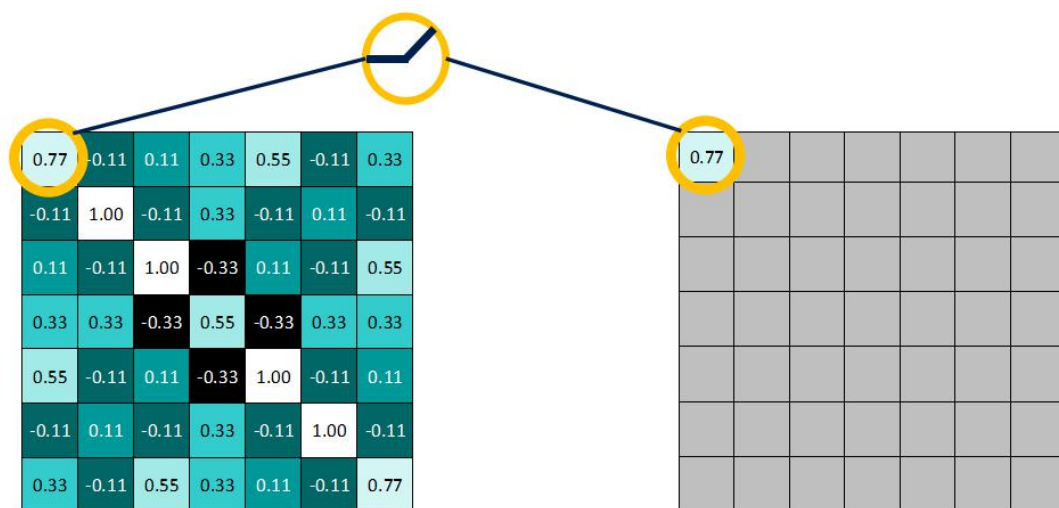


3D体素神经网络-非线性激活层

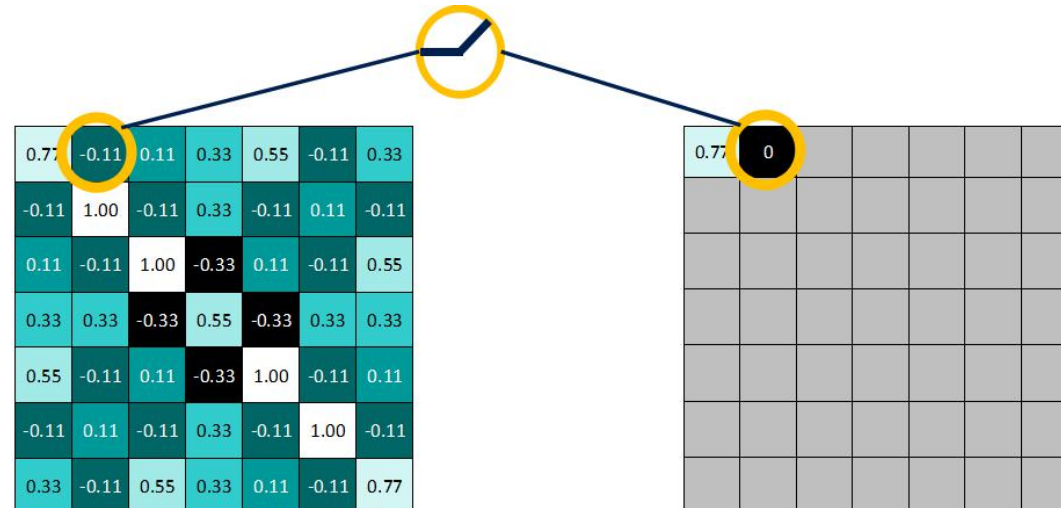
- 卷积层对原图运算多个卷积产生一组线性激活响应，而非线性激活层是对之前的结果进行一个非线性的激活响应。
- 在神经网络中用到最多的非线性激活函数是Relu函数，它的公式定义如下：

$$f(x)=\max(0,x)$$

即，保留大于等于0的值，其余所有小于0的数值直接改写为0。进行特征提取时，为了使得数据更少，操作更方便，就直接舍弃掉那些不相关联的数据



≥ 0 的值不变



< 0 的值一律改写为0

3D体素神经网络-非线性激活层

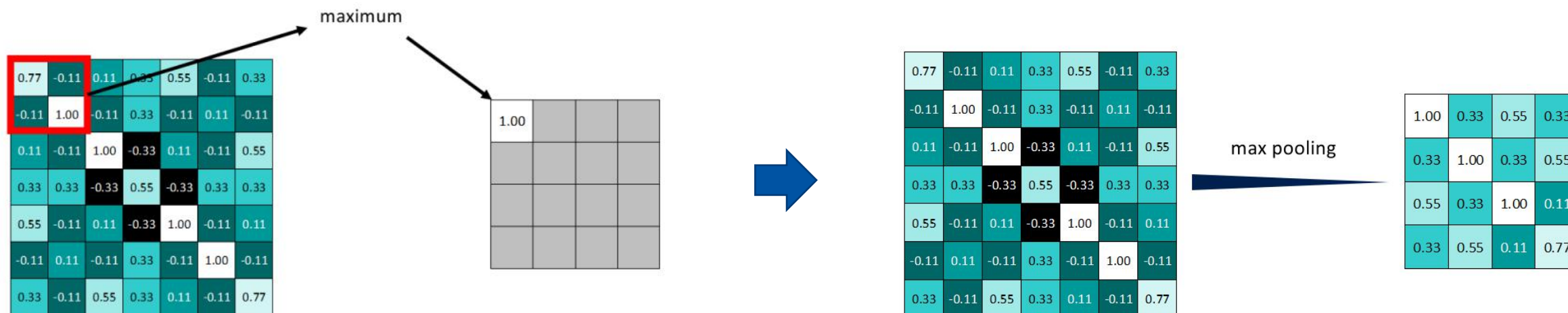
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77



0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

3D体素神经网络-pooling池化层

- 卷积操作后，我们得到了一张张有着不同值的feature map，尽管数据量比原图少了很多，但还是过于庞大，因此接下来的池化操作就可以发挥作用了，它最大的目标就是减少数据量。
- 池化分为两种，Max Pooling 最大池化、Average Pooling平均池化。顾名思义，最大池化就是取最大值，平均池化就是取平均值。
- 拿最大池化举例：选择池化尺寸为2x2，因为选定一个2x2的窗口，在其内选出最大值更新进新的feature map。



得到池化后的feature map。可明显发现数据量减少了很多

3D体素神经网络-pooling池化层

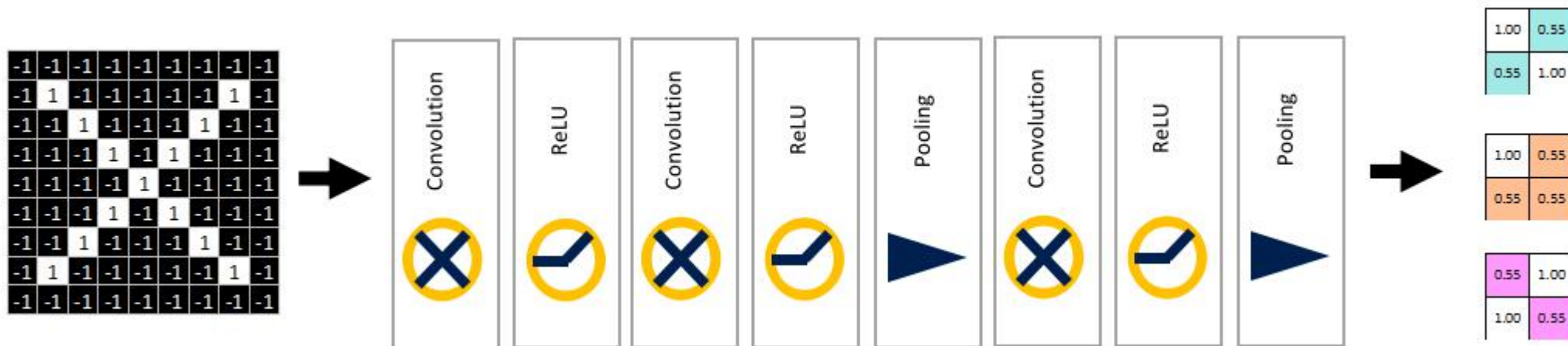
- 这里就介绍了CNN的基本配置---卷积层、Relu层、池化层



- 也可以自行添加更多的层以实现更为复杂的神经网络。

3D体素神经网络-全连接层

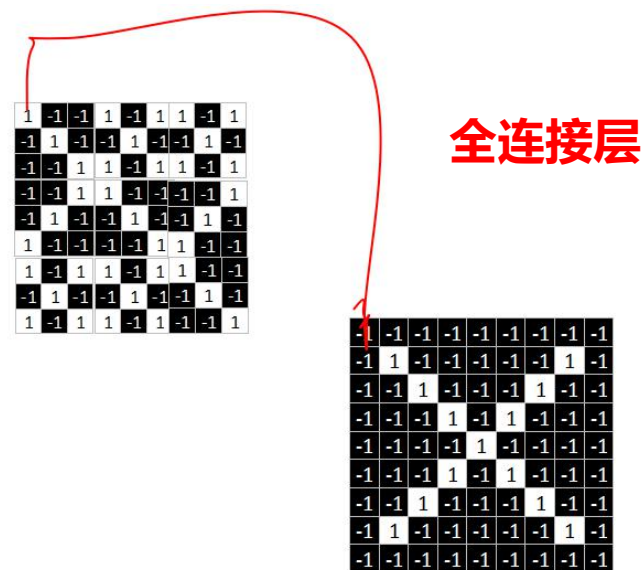
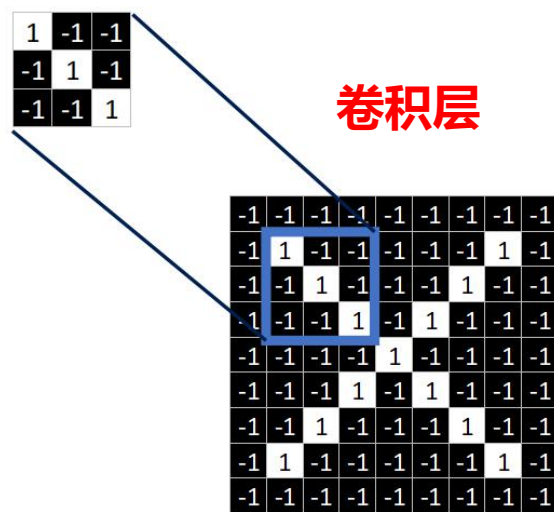
- 原图片尺寸为9X9，在一系列的卷积、relu、池化操作后，得到尺寸被压缩为2X2的三张特征图。



- 想想我们最初和最终的目的到底是什么？是对这张照片进行识别，识别它到底是X还是O呢（其实也算是对它进行一个二分类）。那我们得到的是什么？是一个2X2的矩阵，好像和分类并没有什么关系

3D体素神经网络-全连接层

- 全连接层要做的，就是对之前的所有操作进行一个总结，给我们一个最终的结果
- 全连接层，顾名思义就是全部都连接起来。
 - **卷积层**：采用的是“局部连接”的思想，是用一个3X3的图与原图进行连接操作，很明显原图中只有一个3X3的窗口能够与它连接起来。
 - **全连接层**：拿9X9的输入原图做栗子，要进行全连接的话，那权值参数矩阵应该也是9x9才对，保证每一个值都有对应的权值参数来运算。（二者坐标直接一一对应）

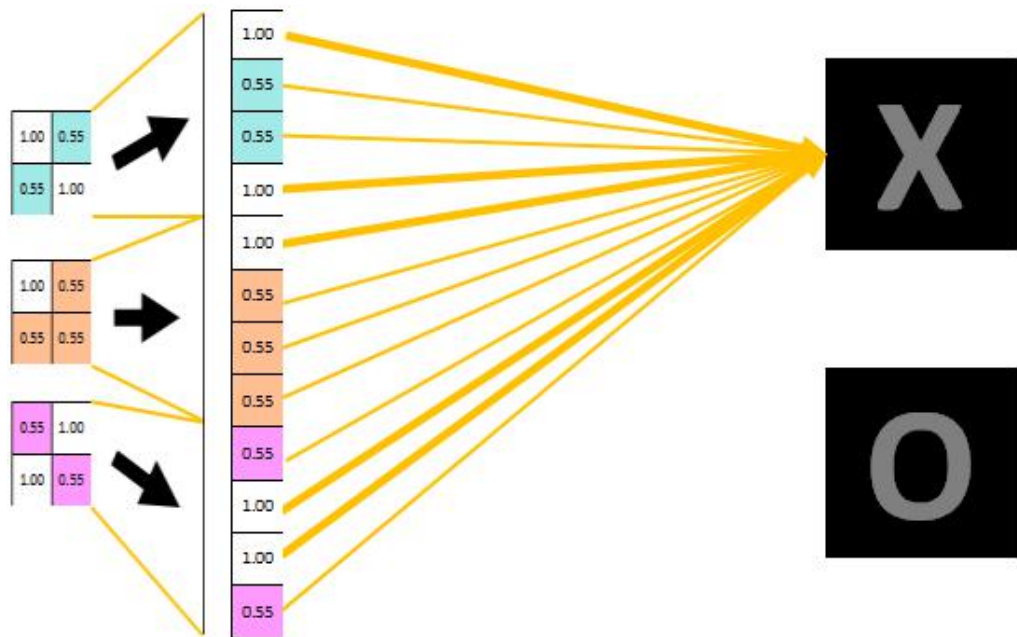


3D体素神经网络-全连接层

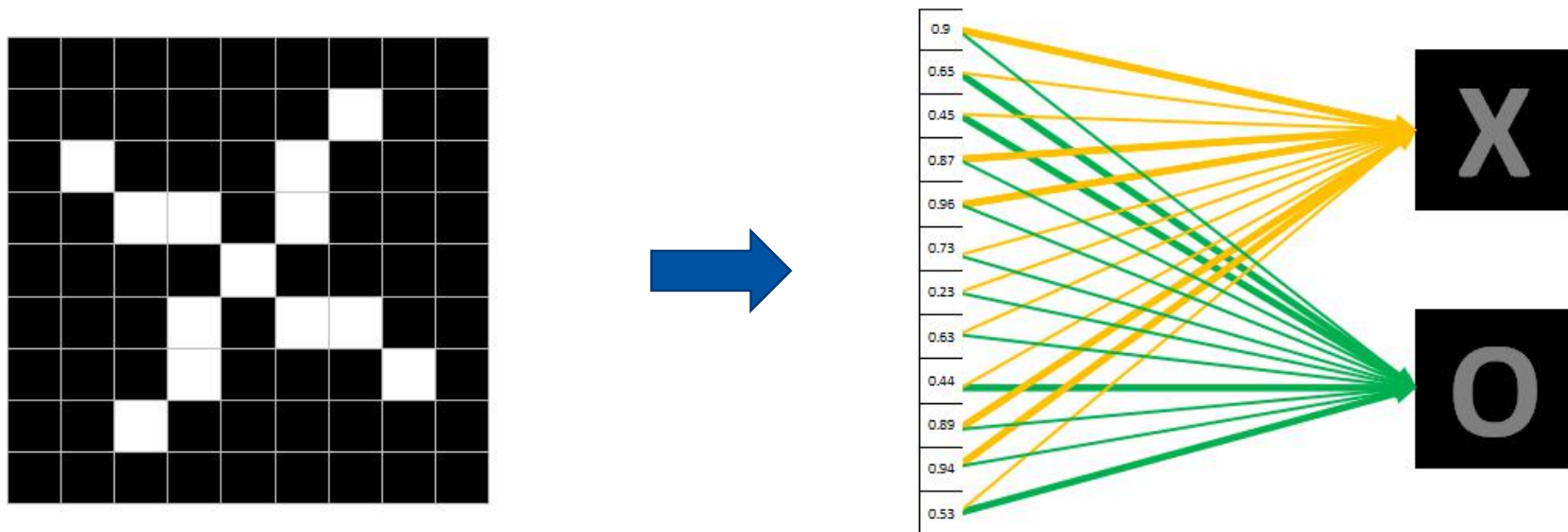
- 得到了2X2的特征图后，对其应用全连接网络，再全连接层中有一个非常重要的函数----Softmax，它是一个分类函数，输出的是每个对应类别的概率值。比如：

【0.5, 0.03, 0.89, 0.97, 0.42, 0.15】就表示有6个类别，并且属于第四个类别的概率值**0.97**最大，因此判定属于第四个类别。

注意:本例中因为只有两个类别X和O，而且数据量到此已经非常少了，因此直接将三个特征图改变维度直接变成一维的数据。（相当于全连接层的每个参数均为1）

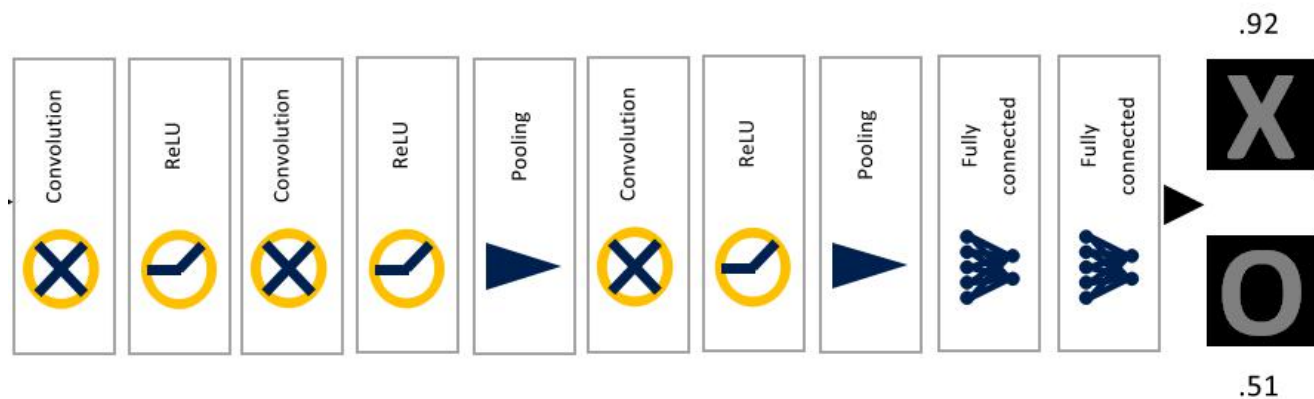


3D体素神经网络-全连接层



0.9表示极其大可能是X，因此对应到X的黄色线条比对应到O的绿色线条要粗很多很多。

我们对结果进行统计分析后可判断这张图片里的字母为X。



3D体素神经网络-神经网络的训练与优化

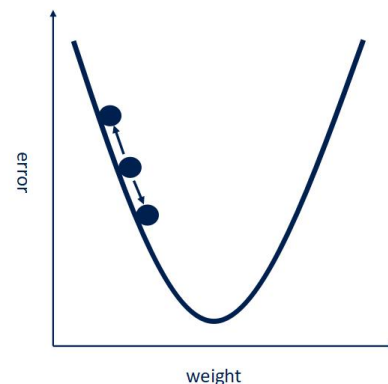
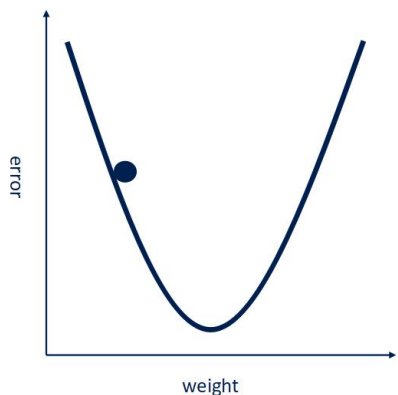
- 神经网络到底需要训练什么呢？训练的就是那些卷积核（filter）
- 针对这个识别X的例子，我们可以人为定义三个3X3的卷积核，便可实现对X的特征提取。
- 但是在实际运用中，比如识别手写字母，几乎不可能存在标准的写法，每个人的字迹都完全不同，经过成千上万的训练集来训练，每一次加入新的数据，都有可能对卷积核里的值造成影响。

BackProp反向传播算法

1、训练前，我们定义一个大小为3X3的卷积核，那么里面具体的值是多少，我们都不知道，但又不能为0吧，所以就用随机初始化法来进行赋值。

2、卷积神经网络便可以开始工作了，输入一张带有标签的图片（假设图片内容是字母X）。经网络识别后判断是X的概率为0.3。

3、一种简单定义误差error的计算公式为 $\text{error} = (\text{result} - \text{label})^2$
训练的终极目的就是使得这个误差最小，常用的方法是 梯度下降法





谢谢