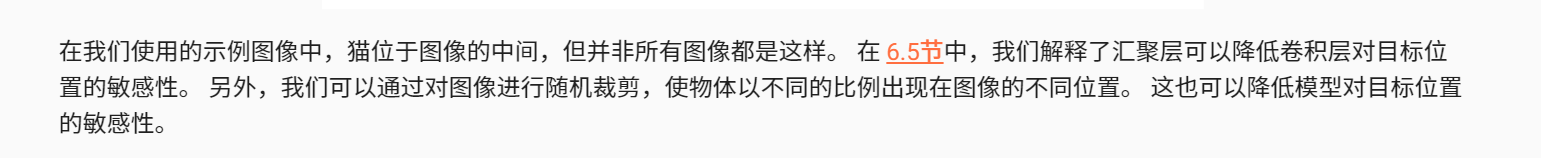
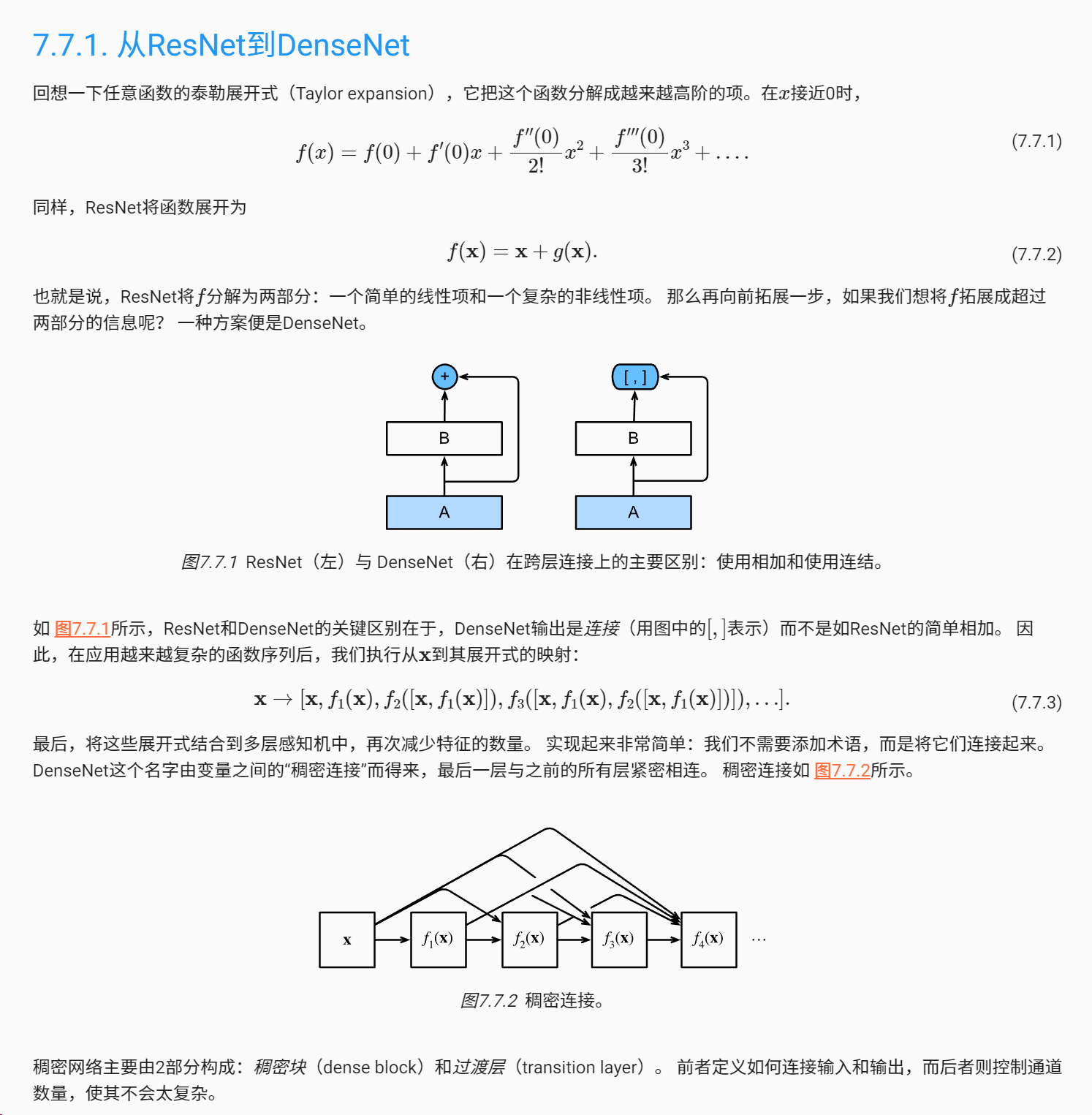
# 汇聚层

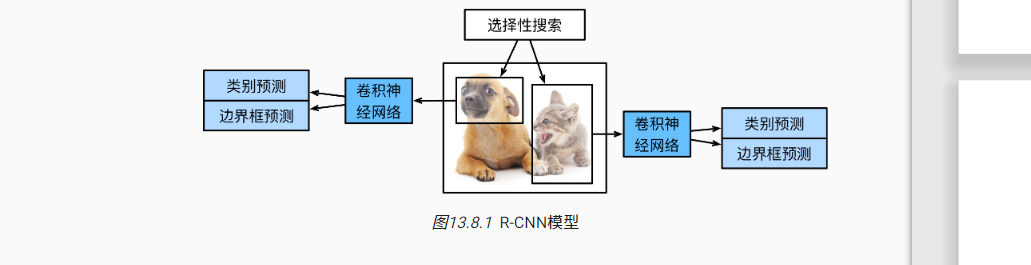


# DenseNet

利用了类似泰勒公式的原理，将我们需要拟合的函数f(x)分解成了两个部分，Dense就是这样定义的，他只是类似泰勒，但不是泰勒。

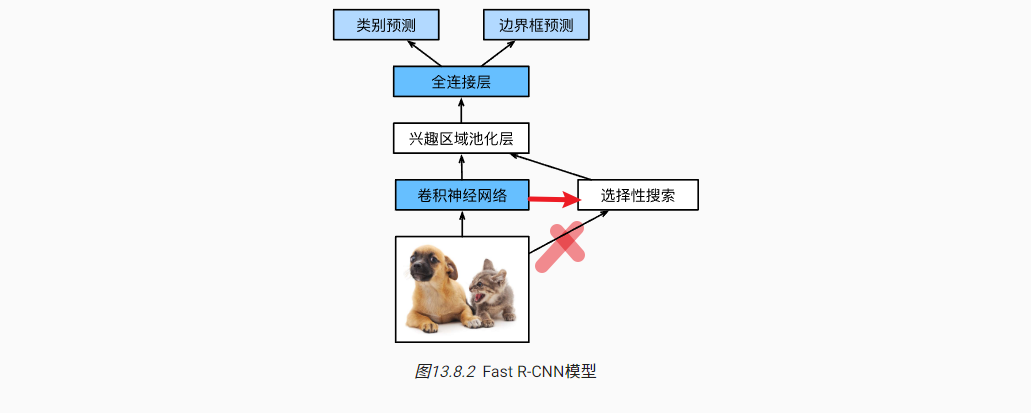


# CNN(region-based CNN)



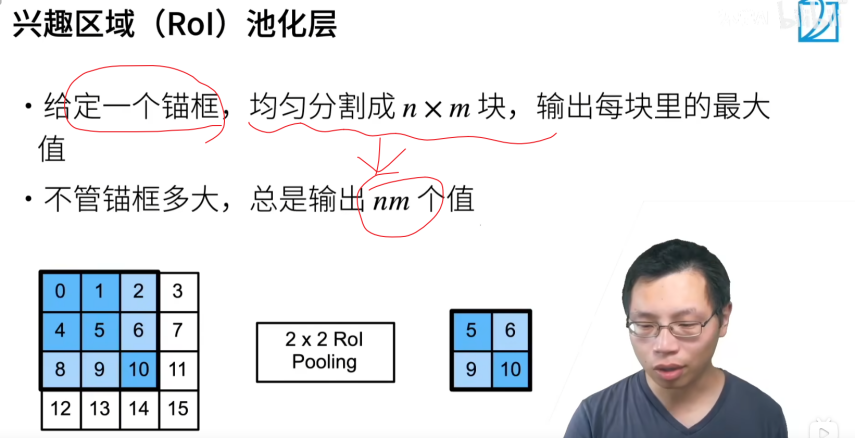
先利用SS对图片进行提取锚框,然后将所有的锚框整合成一个Batch Size,再走CNN

## Fast R-CNN

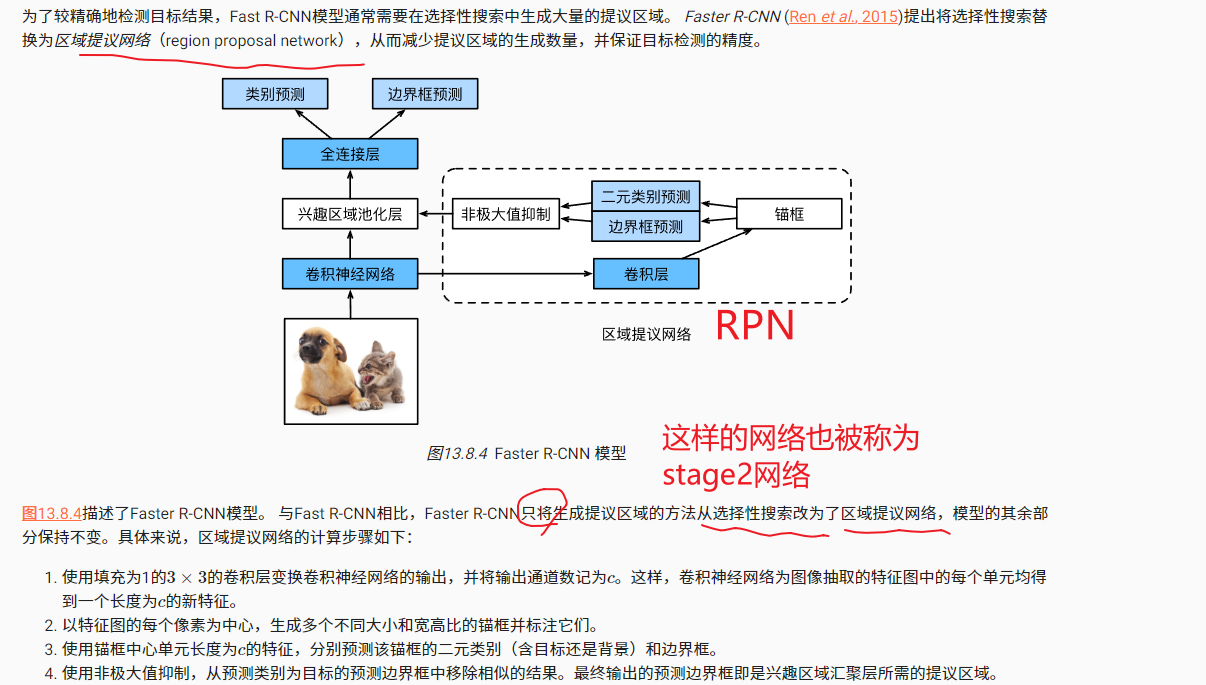


直接将图片输入CNN提取特征,然后对特征图进行SS(选择性搜索),再进入兴趣区域池化层

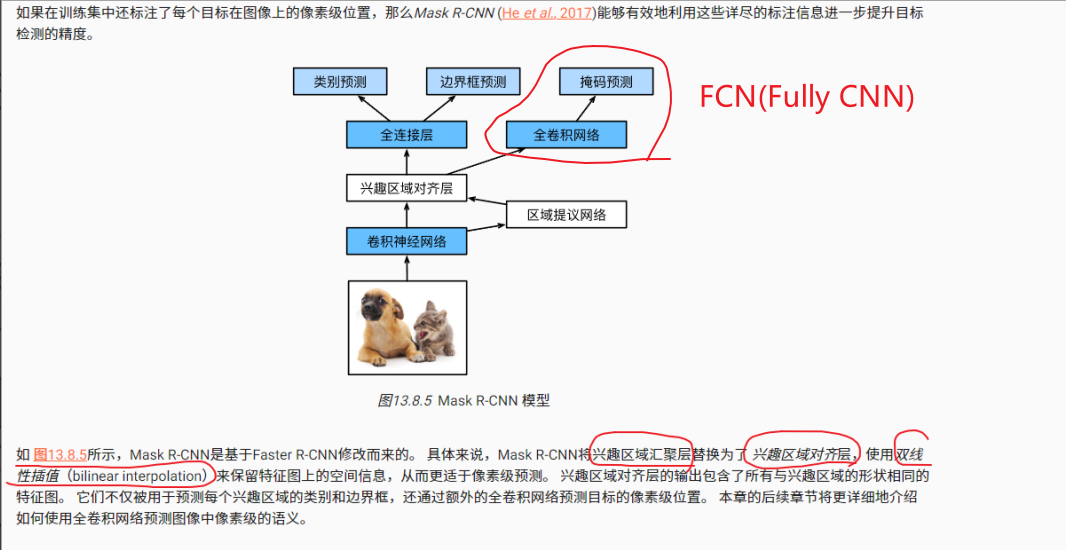
### 兴趣区域池化层



## Faster R-CNN

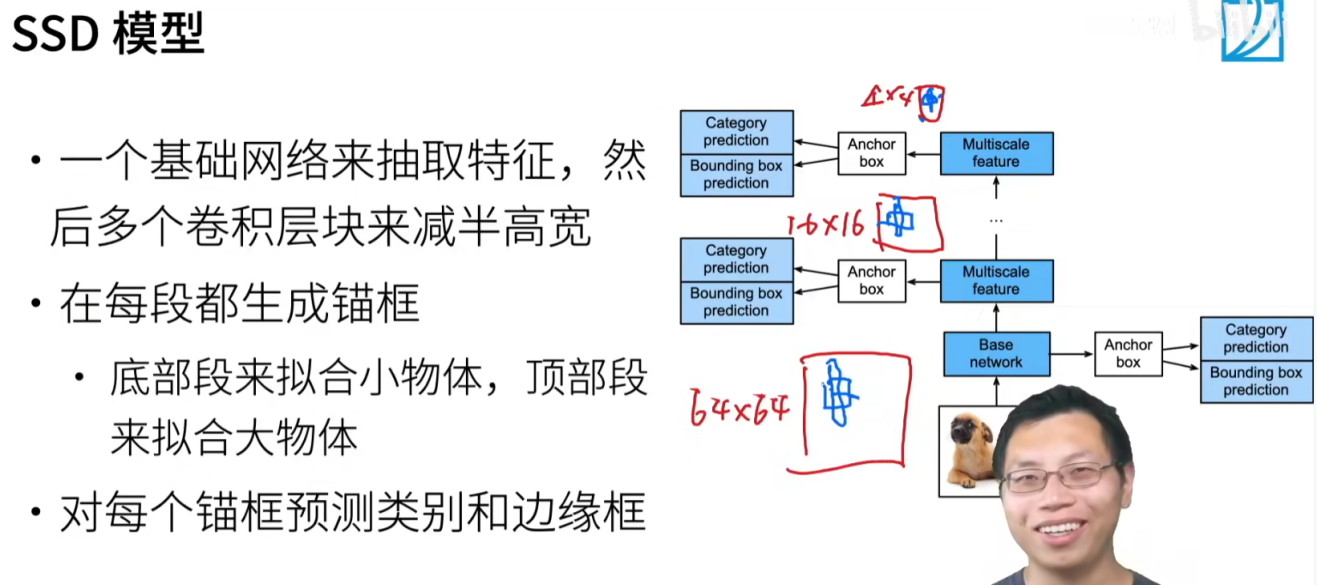


## Mask R-CNN



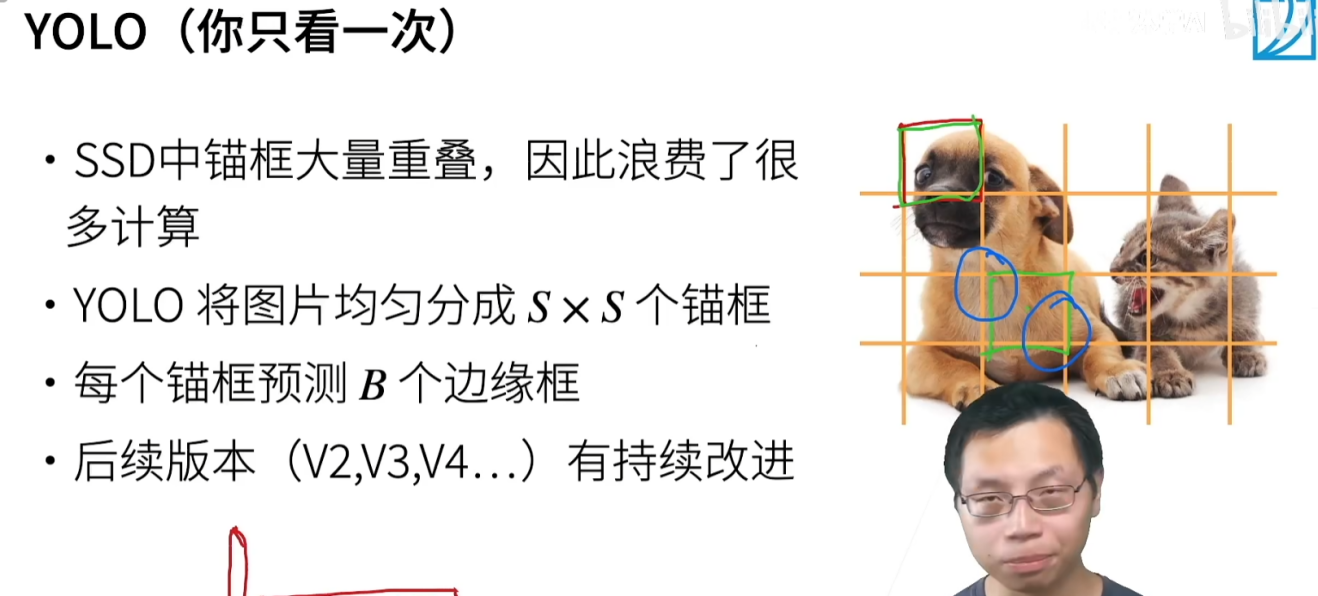
应用FCN

# SSD

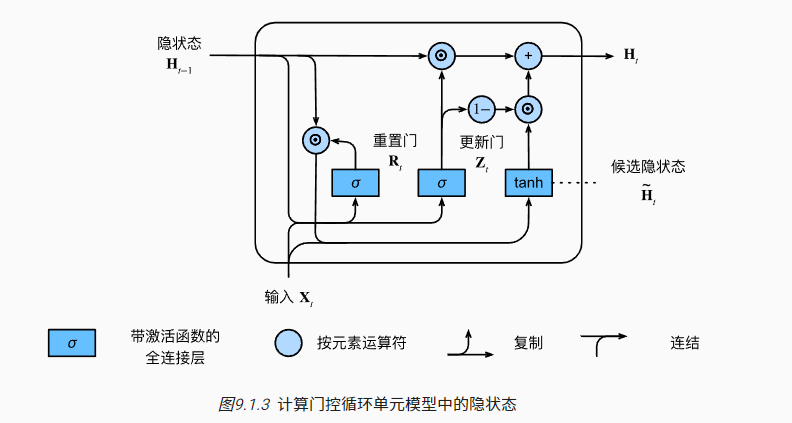


越顶部,感受野越大,用来识别大物体

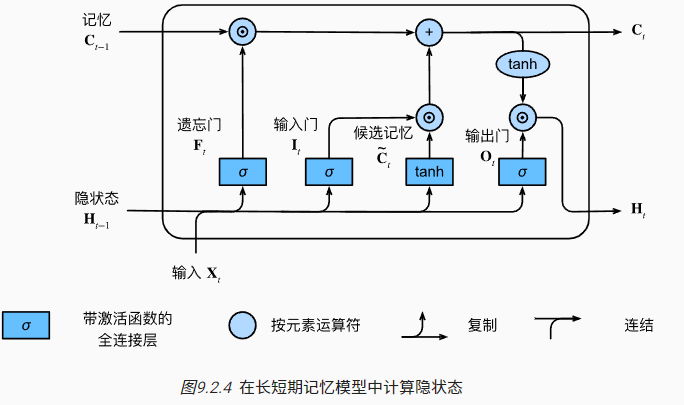
# YOLO



# GRU



# LSTM



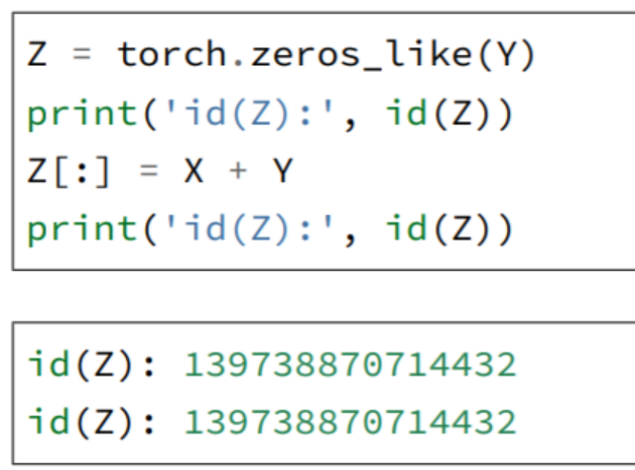
笔记表示

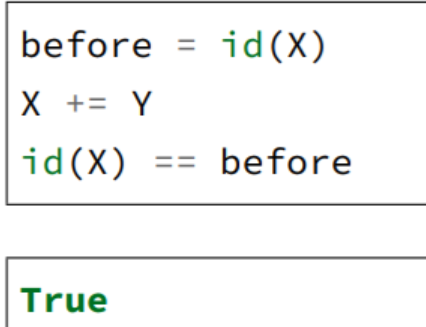
X为张量，向量，x为标量

# 第二章

## 2.1数据操作

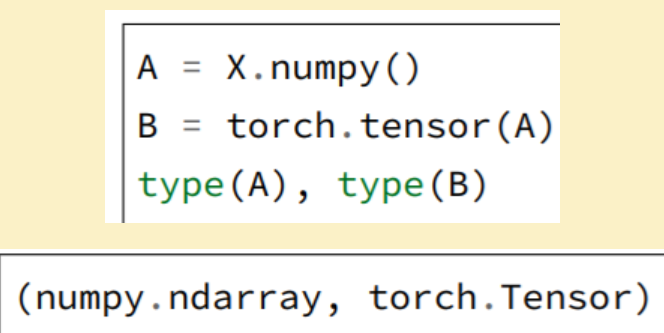
### 节省内存





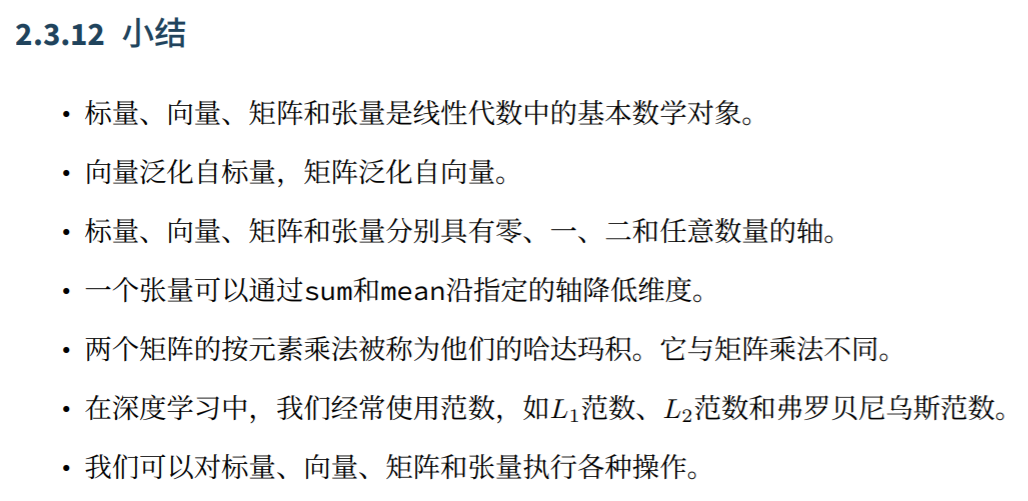
## 2.2数据预处理

### 转换为Ndarray



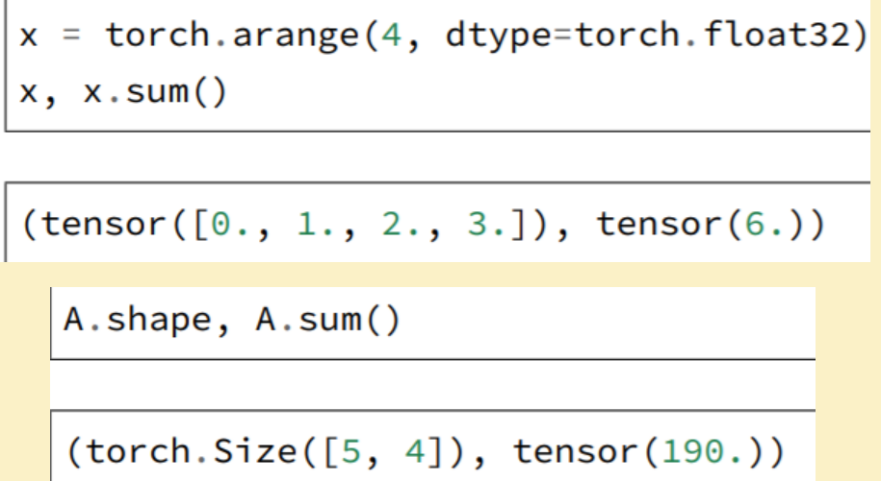
## 2.3线性代数

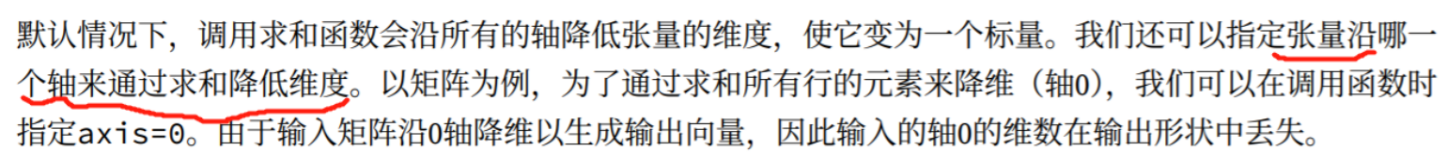
### 小结

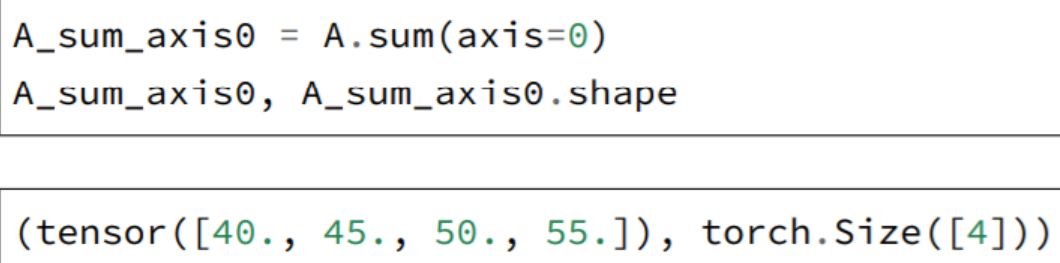


### 降维

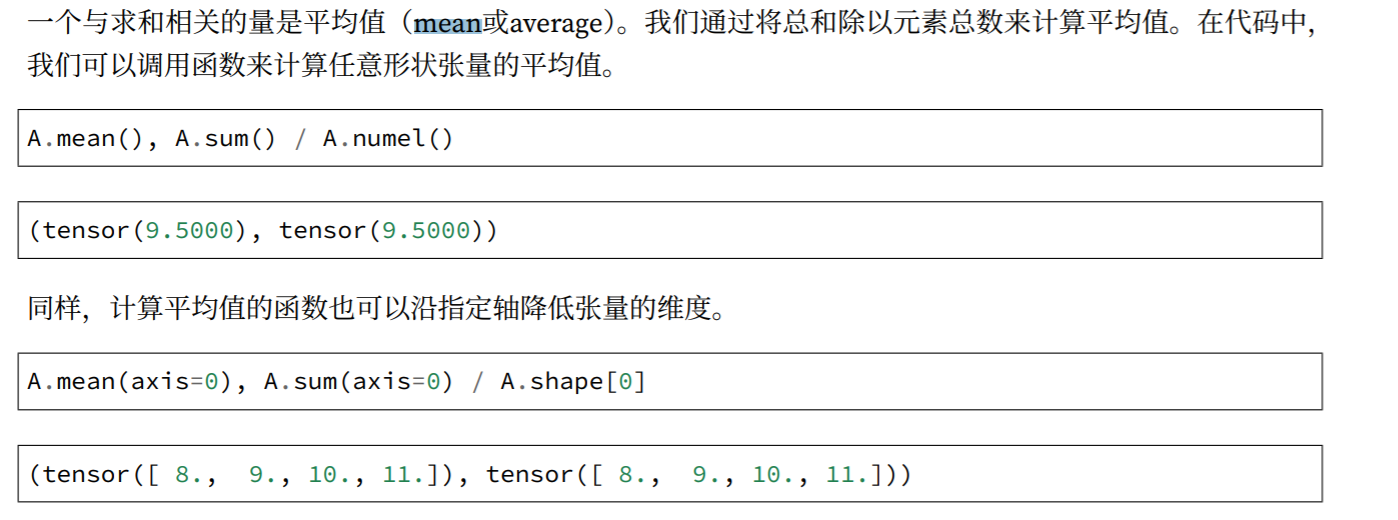
#### 求和降维 X.sum()



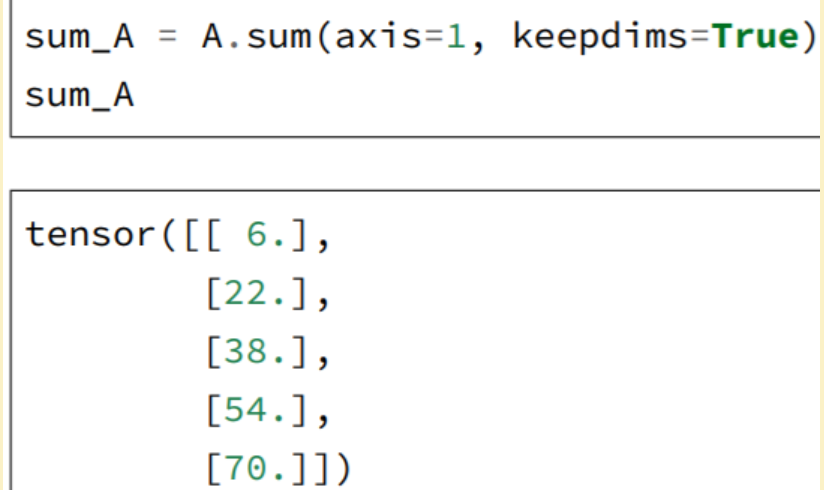




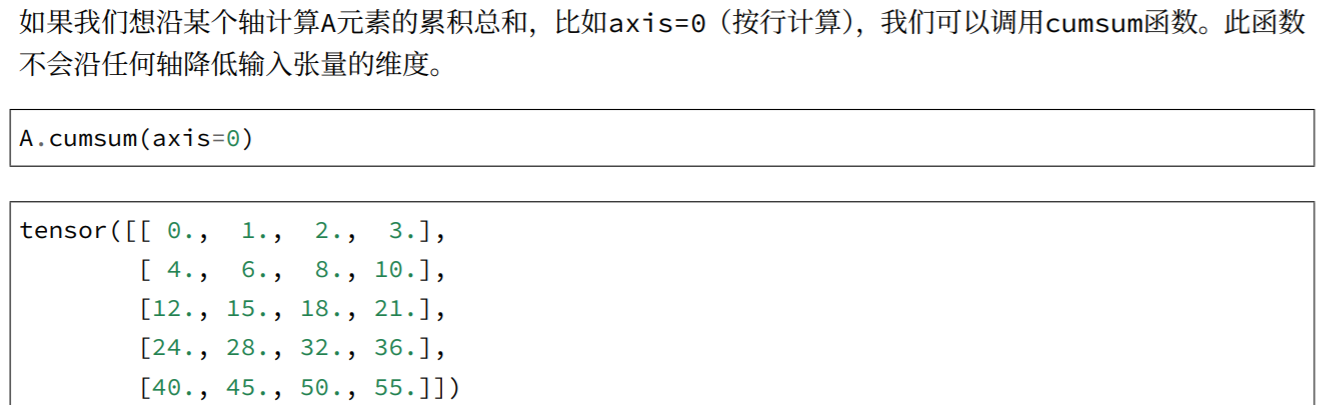
#### 平均值降维Mean（X）



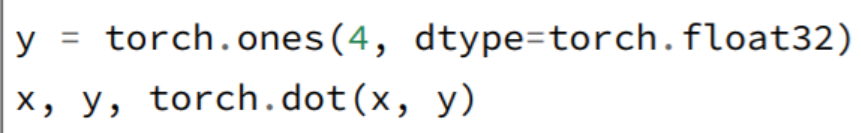
#### ⾮降维求和

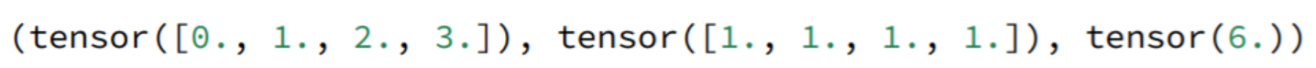


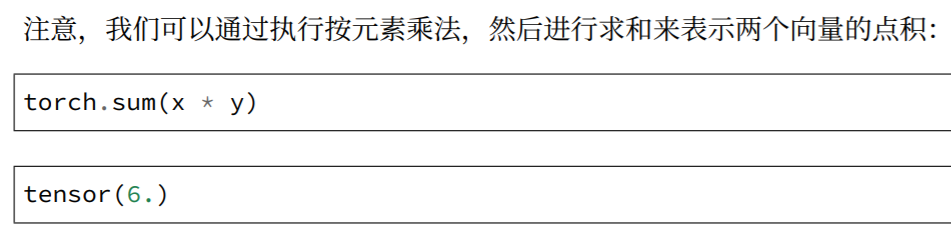
、



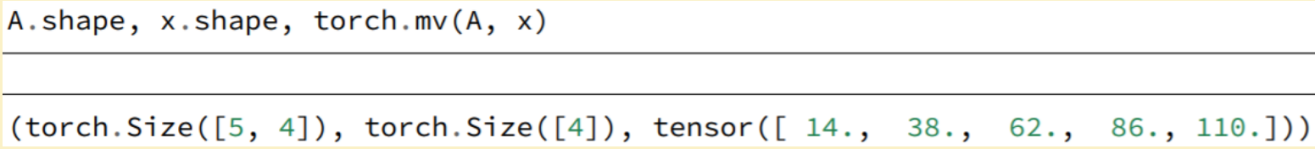
### 点积 torch.dot(X1,X2)





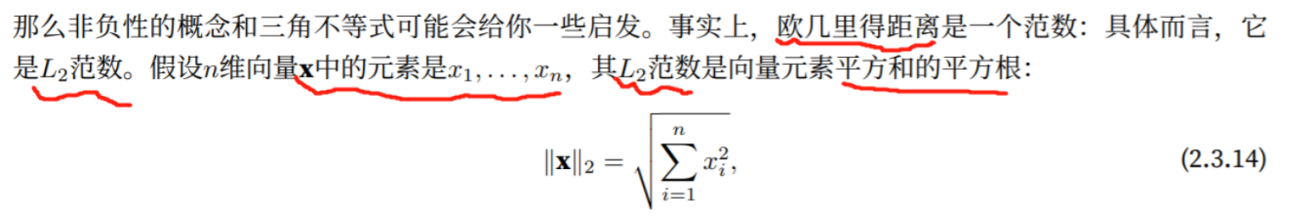


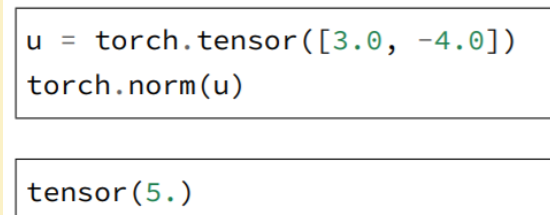
### 矩阵向量积torch.mv(X1,X2) X1为矩阵，X2为向量



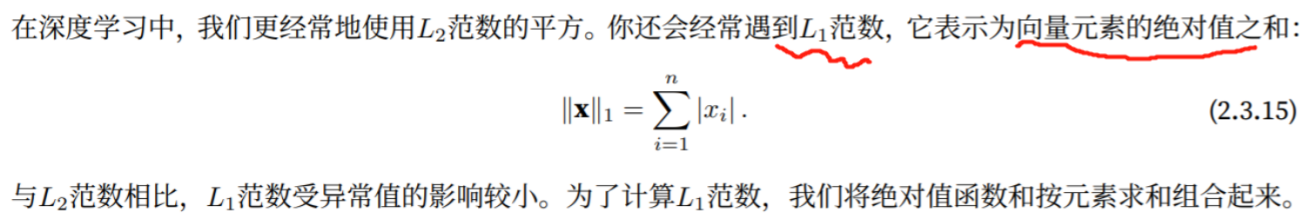
### 范数 ⼀个向量的范数告诉我们⼀个向量有多⼤。

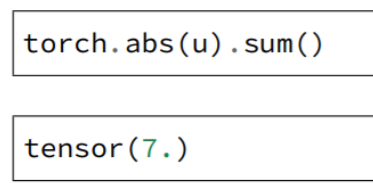
#### L2范数 torch.norm(X) 向量



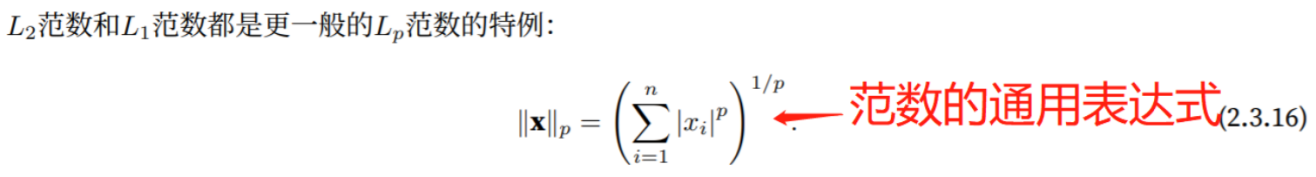


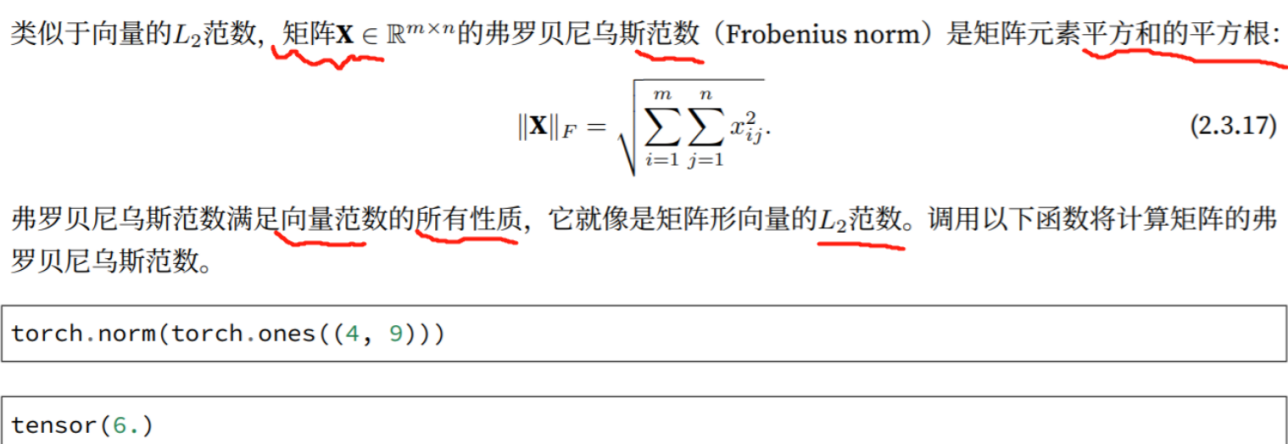
#### L1范数 torch.abs(X).sum() 向量





#### 矩阵 弗罗贝尼乌斯范数 torch.norm(X) 矩阵





# 学习总结：

## 张量相乘

Tensor\_A.shape = (a, b)

Tensor\_B.shape = (b, c)

Tensor\_A x Tensor\_B = Tensor (a, c)

## 深度学习的原理

假设我们有一个loss表示为：

^2

我们想让y获得最小值，就进行，可以得到，我们对其求解，令就能得到y的最小值。因为导数能显示y的增长率，当时，增长率为0，就是最低处。

同理的，如果我们的Loss表示为：



分别进行，，，令，，就可以求出的最小值。

## 常用函数

tensor : class

### tensor.detach()分离计算

分离计算，梯度会丢弃其计算信息

### torch.matmul()两个张量的矩阵乘积。

- 如果两个张量都是一维的，将返回点乘（标量）。

- 如果两个参数都是二维的，则返回矩阵-矩阵乘积。

- 如果第一个参数是一维的，第二个参数是二维的。

为了矩阵乘法的目的，会在其维度上预加1。

矩阵乘法后，预置的维度将被移除。

- 如果第一个参数是二维的，第二个参数是一维的。

将返回矩阵-向量的乘积。

- 如果两个参数至少都是一维，并且至少有一个参数是

N维（其中N>2），那么将返回一个分批的矩阵乘法。 如果第一个

参数是一维的，则在其维度上预加1，以便进行矩阵乘法的目的，在其维度上加一个1，然后再删除。 如果第二个参数是一维的，则在其维度上添加一个1会被添加到它的维度上，以便进行分批矩阵乘法，之后再删除。