代码内容审核建议 11 （2.6.1~2.6.6,2.8.1,~2.8.2）

1. **2.6 可以添加对 pytorch 的安装以及 gpu 库的简要介绍**

   2.6.1 先是引入深度学习，2.6.1.1 介绍了借助深度学习框架 pytorch 介绍了张量 tensor，由于 pytorch 的使用通常要借助 GPU 的 cuda 来加速并行运算，且安装较为复杂，**建议添加一个小章节来补充介绍**

   且 pytorch 的主要版本和 cuda 版本，python 版本的依赖关系对后续使用有很多影响，此处**最好点明主要的软件版本**

   参考网站

Pytorch 最新版本安装官网 https://pytorch.org/get-started/locally/ Start Locally | PyTorch

Pytorch 旧版本 Previous PyTorch Versions | PyTorch

Cuda 最新版本 CUDA Toolkit 11.7 Update 1 Downloads | NVIDIA Developer

cuda 所有版本汇总 CUDA Toolkit Archive | NVIDIA Developer

2. **2.6.1 整体完成度较高，最后一段有对 pytorch 和 tensorflow 以及 keras 的介绍，其实还可以添加下对国产的深度学习框架 paddle-paddle 的介绍，目前来说也是很受欢迎的。另外，pytorch 最开始受欢迎的原因是支持动态图的构建，后续 tensorflow 和 keras 合并后，TF2.x 版本后也是支持动态图构建了，可以进行一些调研和补充**

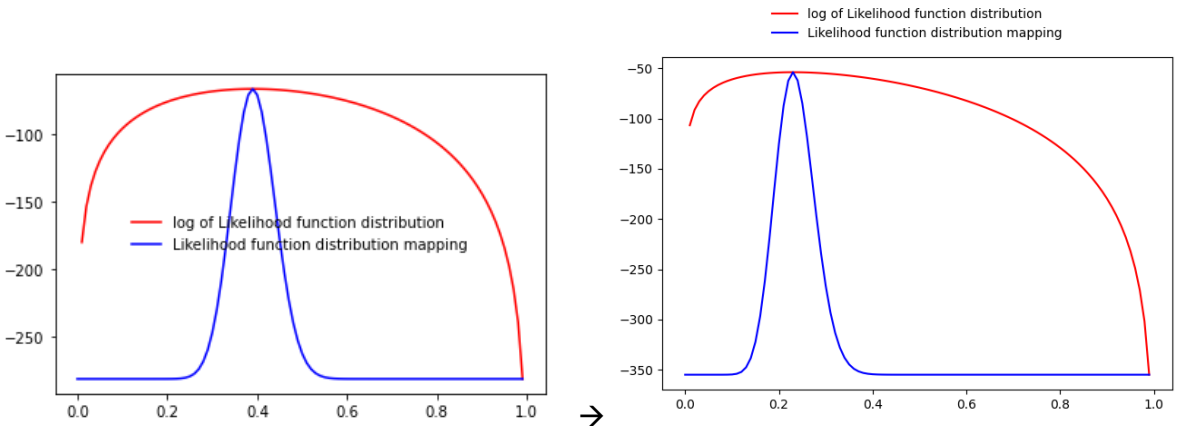
飞桨 PaddlePaddle-源于产业实践的开源深度学习平台

https://www.paddlepaddle.org.cn/

3. **2.6.2.3 最后的图片中 legend 挡住图片不美观，修改代码**

将倒数第二行代码“`ax.legend(loc='best',frameon=False)`**”改为**
“ax.legend(bbox_to_anchor=(0.5, 1), loc=8, borderaxespad=1, frameon=False)”

下面是修改前后的比较

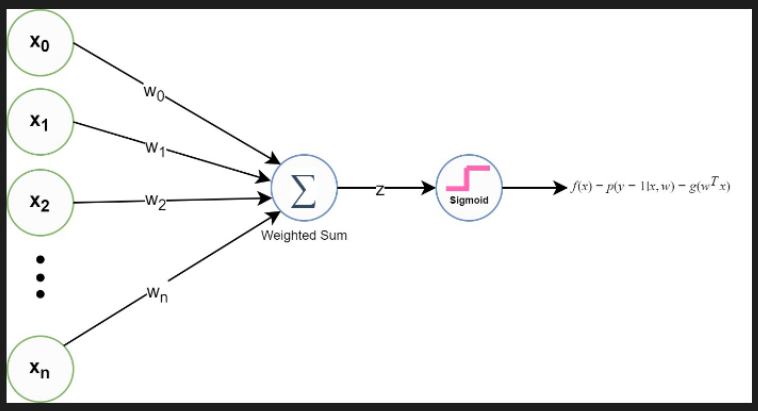


4. 2.6.2.4 中 alpha 学习率调整公式中，**正文中未对 i 和 j 做出解释**【见红色方框标注】

后面的 stocGradAscent1 代码中，可以推断，i 是和训练迭代次数有关的变量，j 是和数据维度有关的变量

2.6.2.4 逻辑回归(Logistic Regression, LR)

逻辑回归模型为：$p(x)=\sigma(t)=\frac{1}{1+e^{-t}}=\frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_2 x_n)}}=\frac{1}{1+e^{-w^T x}}$，其中$\sigma(t)$就是sigmoid函数(在深度学习中用于激活函数)，sigmoid函数可以将线性模型$f(x)=\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_2 x_n=w^T x$，归一化到[0,1]之间。而因为为二元分类，类标只有两类值1和0，特征值经过加权和（参数全部初始化为1）喂入sigmoid函数后所得到的值位于[0,1]区间，则该值与类标的差值和就为逻辑回归的损失函数，通过随机梯度下降法更新初始化的参数（并未使用最大似然估计，可能无法解析求解），最终训练所得的参数应该使得逻辑回归模型的预测结果趋近于类标1或0。逻辑回归模型是线性模型和sigmoid函数的组合，这与解释"反向传播"部分所构建的隐含层和输出层的网络结构是一样的，可以互相印证理解。



随机梯度下降stocGradAscent1方法的定义直接迁移*Machine learning in action*书中的代码，作者解释了优化的随机梯度下降，主要改进包括：1，每次迭代时，调整更新步长alpha值（即是学习率lr），学习率会越来越小，从而缓解系数的高频波动。同时为了避免迭代不断减小至0，约束学习率大于一个稍微大点的常数项，对应代码为`alpha = 4/(1.0+j+i)+0.0001` 。同时，改变样本的优化顺序，即是随机选择样本来更新回归系数。这样可以减少周期性的波动。对应的代码为`randIndex = int(np.random.uniform(0,len(dataIndex)))`。

```
def stocGradAscent1(self,dataMatrix, classLabels, numIter=150):
    from tqdm import tqdm
    '''
    function - 随机梯度下降（优化）
    '''
    m,n = np.shape(dataMatrix)
    weights = np.ones(n)   #initialize to all ones
    for j in tqdm(range(numIter)):
        dataIndex = list(range(m))
        for i in range(m):
```

5. Softmax 函数的这两种表述，事实上是一样的，只是权重 W 和特征 X 的位置问题，**建议只保留一种，**

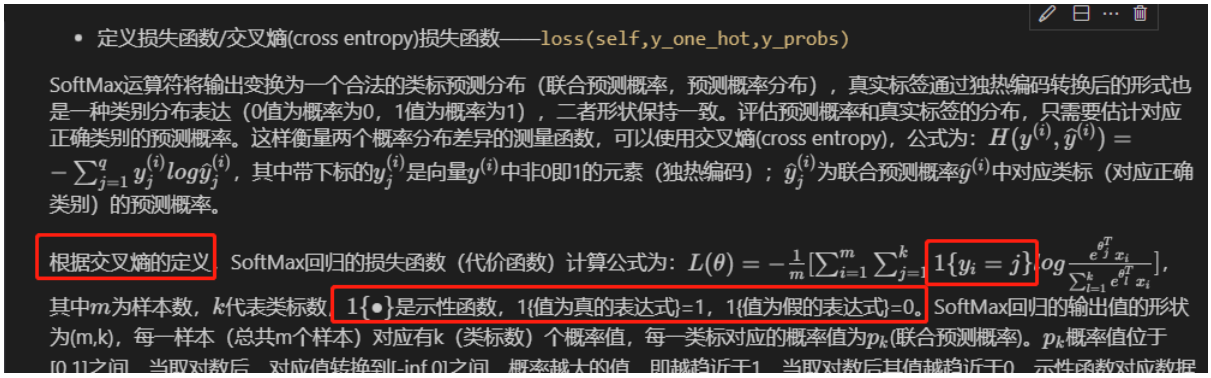
- 定义SoftMax函数——`softmax(self,z)`

SoftMax回归/函数，或称为归一化指数函数，是逻辑回归/函数(Logistic Regression)的一种推广。将一个含任意实数的K维向量z"压缩"到另一个K维实向量$\sigma(z)$中，使得每一个元素的范围都在(0,1)之间，并且所有元素的和为1。该函数的形式通常为：$\sigma(z)_j=\frac{e^{z_j}}{\sum_{k=1}^{K}e^{z_k}}\quad for\quad j=1,\ldots,K$。SoftMax函数实际上是有限项离散概率分布的梯度对数归一化，广泛应用于多分类问题方法中。在多项逻辑回归和线性判别分析中，函数的输入是从K个不同的线性函数得到的结果，而样本向量x属于第j个分类的几率为：$P(y=j|x)=\frac{e^{x^T w_j}}{\sum_{k=1}^{K}e^{x^T w_k}}$，这可以被视作K个线性函数$x\mapsto x^T w_1,\ldots,x\mapsto x^T w_k$，SoftMax函数的复合。亦可以表述为，对于输入数据$\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$，有K个类别，即$y_i\in\{1,2,\ldots,k\}$,那么SoftMax回归主要估算输入数据$x_i$会属于每一类的概率，即$h_\theta(x_i)=\begin{bmatrix}p(y_i=1\mid x_i;\theta)\\p(y_i=2\mid x_i;\theta)\\\vdots\\p(y_i=k\mid x_i;\theta)\end{bmatrix}=\frac{1}{\sum_{k=1}^{K}e^{\theta_k^T x_i}}\begin{bmatrix}e^{\theta_1^T x_i}\\e^{\theta_2^T x_i}\\\vdots\\e^{\theta_k^T x_i}\end{bmatrix}$，其中$\theta_1,\theta_2,\ldots,\theta_k\in\theta$是模型的参数，乘以$\frac{1}{\sum_{k=1}^{K}e^{\theta_k^T x_i}}$是为了让概率位于[0,1]并且概率之和为1，SoftMax回归将输入数据x_i归属于类别j的概率为：$p(y_i=j|x_i;\theta)=\frac{e^{\theta_j^T x_i}}{\sum_{k=1}^{K}e^{\theta_k^T x_i}}$。

不同的作者可能对同一问题的公式表述有所差异，例如不同的变量符号，或者表述规则，例如上述两种阐述中，模型的权值分别被表述为w和θ，亦或者以矩阵的方式表述的方程组等。不管公式的表达方式如何，其核心的算法始终是保持一致的，可以根据自己容易理解的方式来书写表达式。

结合后文中的交叉熵也是采用θ，此处可以只保留第二种

6. 关于交叉熵和 softmax 函数的定义，对于**示性函数1{·}函数，这种表达方式不是很通用常见**



下面这种表示方法更为使用普遍，【建议修改】

损失函数：

衡量期望值与实际值之间的误差的函数。深度学习的目的是最小化损失函数！因为期望值与实际值之间误差越小所得的模型的权重系数越好！

交叉熵函数（该函数越小预测结果越正确）：

交叉熵就是用来判定实际的输出与期望的输出的接近程度！！！交叉熵刻画的是实际输出（概率）与期望输出（概率）的距离，也就是交叉熵的值越小，两个概率分布就越接近！！！所以可以用交叉熵来作为损失函数！！！以下公式中的 H(p,q)就是交叉熵！

第一种交叉熵函数：

$$H(p,q) = -\sum_{x} p(x) log q(x)$$

第二种交叉熵函数：

$$H(p,q) = -\sum_{x} (p(x) log q(x) + (1-p(x)) log(1-q(x)))$$

其中,p(x)是期望输出，q(x)是实际输出。

https://blog.csdn.net/jiao_mrswang/article/details/97369078

7. Fashion-MNIST 数据集介绍中将“物品”修改为**“种类”**

5）PyTorch_SoftMax回归多分类，用于图像数据集Fashion-MNIST

- 图像数据集Fashion-MNIST

Fashion-MNIST包括60,000个例子的训练集和10,000个例子的测试集。每个示例都是一个28 × 28灰度图像，总共784像素，与来自10个类的标签相关联，可用于替代原始MNIST手写数字数据集，对机器学习算法进行基准测试。 图像的像素都有一个与之相关联的像素值，表示该像素的明度和暗度，数字越大表示越暗。像素值为0到255的整数。训练和测试数据集有785列，第一列由标签组成，表示服装的物品，其余的列包含关联图像的像素值。如果要定位一幅图像一个像素的位置，可以应用$x = i \times 28 + j$定位，其中i, j是0到27之间的整数，像素(x)位于一个28 × 28矩阵的第i行和第j列。

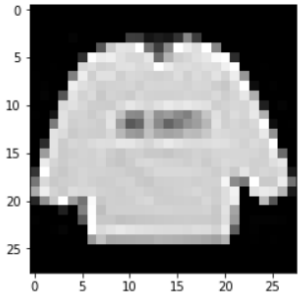
8. 打印展示 Fashion-MNIST 数据集时，可以转换为**灰度显示**，这样看着自然点

在 imshow 函数中添加, cmap="gray" 或者 cmap="Greys"

```
def fashionMNIST_show(imgs,labels,figsize=(12, 12)):
    import matplotlib.pyplot as plt
    from IPython import display
    import matplotlib_inline.backend_inline
    '''
    function - 打印显示Fashion-MNIST数据集图像
    '''
    matplotlib_inline.backend_inline.set_matplotlib_formats('svg')#以svg格式打印显示
    _, axs=plt.subplots(1, len(imgs), figsize=figsize)
    for ax,img,label in zip(axs,imgs,labels):
        ax.imshow(img.view((28,28)).numpy())
        ax.set_title(label)
        ax.axes.get_xaxis().set_visible(False)
        ax.axes.get_yaxis().set_visible(False)
    plt.show()
imgs_len=10
X=[mnist_train[i][0] for i in range(imgs_len)]
y=[mnist_train[i][1] for i in range(imgs_len)]
fashionMNIST_show(imgs=X,labels=fashionMNIST_label_num2text(y),figsize=(12, 12))
```

```
plt.imshow(image.squeeze(), cmap="gray")
print(label)
```

```
2
```




```
if one_channel:
    plt.imshow(npimg, cmap="Greys")
else:
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
```

后续代码也使用了类似的方法，

9. **多次定义相同函数**: flattenLayer 函数出现**多次**在一个章节中**定义**的问题，代码中可以多次展示，书籍中，最好一个章节，只定义一次会更简洁吧

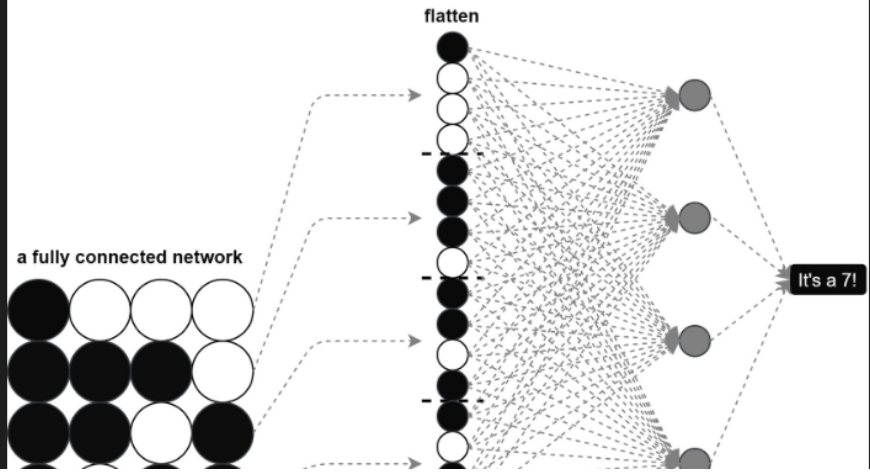
```
#将每批次样本X的形状转换为(batch_size,-1)
class flattenLayer(nn.Module):
    def __init__(self):
        super(flattenLayer,self).__init__()
    def forward(self,x):
        return x.view(x.shape[0],-1)
```

10. **2.6.3.1 的插图和介绍需要修改一下**

2.6.3.1 卷积神经网络(Convolutional neural network, CNN)—— 卷积原理与卷积神经网络

在阅读卷积神经网络(CNNs)之前，如果已经阅读'卷积'，'计算机视觉，特征提取和尺度空间'等部分章节，可以更好的理解CNNs。其中'尺度空间'的概念，通过降采样的不同空间分辨率影像（类似池化，pooLing）和不同σ值的高斯核卷积(即卷积计算或称为互相关运算)，来提取图像的概貌特征，这与CNNs的多层卷积网络如出一辙，只是CNNs除了卷积层，还可以自由加入其它的数据处理层，提升图像特征捕捉的几率。

一幅图像的意义来自于邻近的一组像素，而不是单个像素自身，因为单个像素并不包含关于整个图像的信息。例如下图(引自*PyTorch Artificial Intelligence Fundamentals*)很好的说明了这两者的差异。一个全连接的神经网络（密集层，dense layer），一层的每个节点都连接到下一层的每个节点，并不能很好的反映节点之间的关系，也具有较大的计算量；但是卷积网络利用像素之间的空间结构减少了层之间的连接数量，显著提高了训练的速度，减少了模型的参数，更重要的是反映了图像特征像素之间的关系，即图像内容中各个对象是由自身与邻近像素关系决定的空间结构（即特征）所表述。

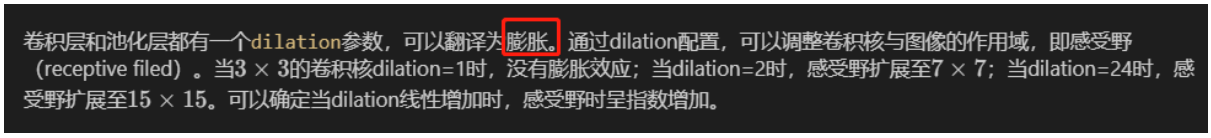


CNN 和全连接神经网络相比，最大的优势是卷积核，可以共享参数，来介绍计算量，且可以在一定程度上保持输入的空间特征，尤其是对图片这种输入。此部分的图片没有体现卷积层的卷积核共享参数，且提取的特征最终 flatten 成向量和全连接层连接，也有点奇怪，建议这部分重新修改下。

11. **W0,W1 没有采用 markdown 公式上下标写法**

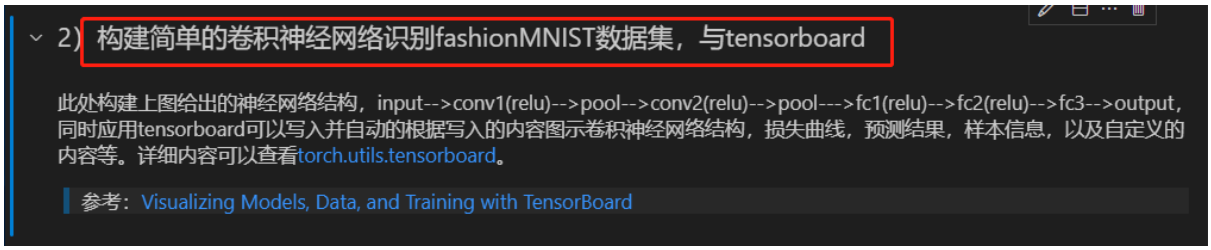
分卷积核对应空值（没有图像/数据）。同时要保持四周填充的行列数相同，通常使用奇数高宽的卷积核；三是，如果是对图像数据实现卷积，通常包括1个单通道，或3个多通道的情况。对于多通道的计算是可以配置不同的卷积核对应不同的通道，各自通道分别卷积后，计算和（累加）作为结果输出。同时可以增加并行的新的卷积计算，获取多个输出，例如下图的Filter W0，和Filter W1的(3 × 3)卷积核，W0和W1各自包含3个卷积核对应3个通道输入，并各自输出。

12. Dilation 通常翻译为**扩张率或者膨胀系数**（dilation rate），和卷积一起使用的时候，通常翻译为**空洞卷积（dilated convolution）**

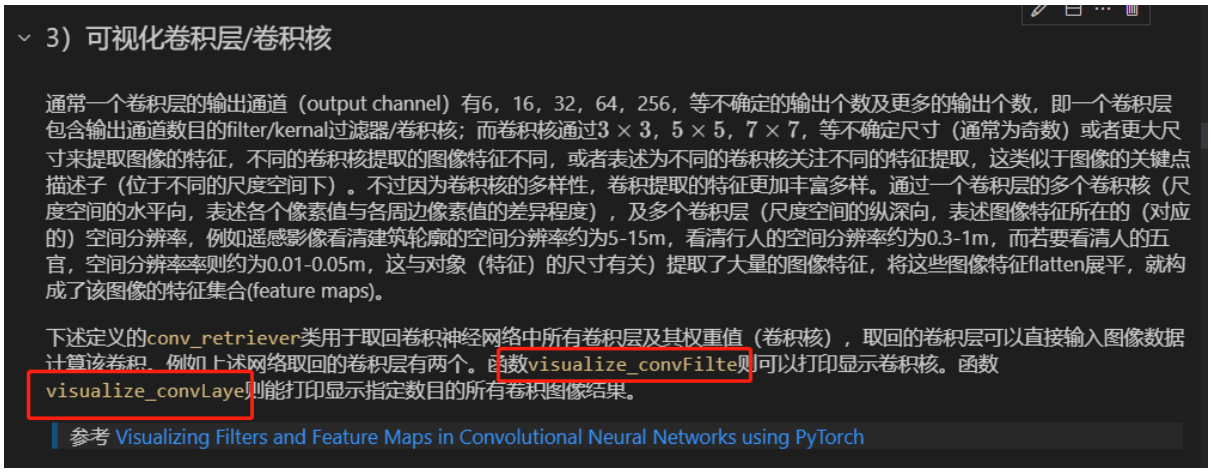


13. 此处的 h_w 并不在 conv 和 pool 中出现，可以删去调整

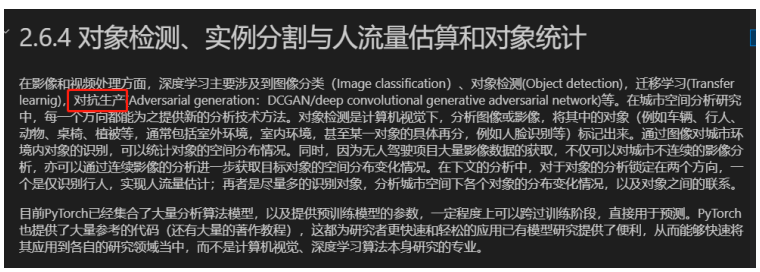
14. 此处题目需要调整下，可以改为，**基于 fashionMNIST 数据集构建简单的卷积神经网络识别任务，并用 tensorboard 可视化展示**



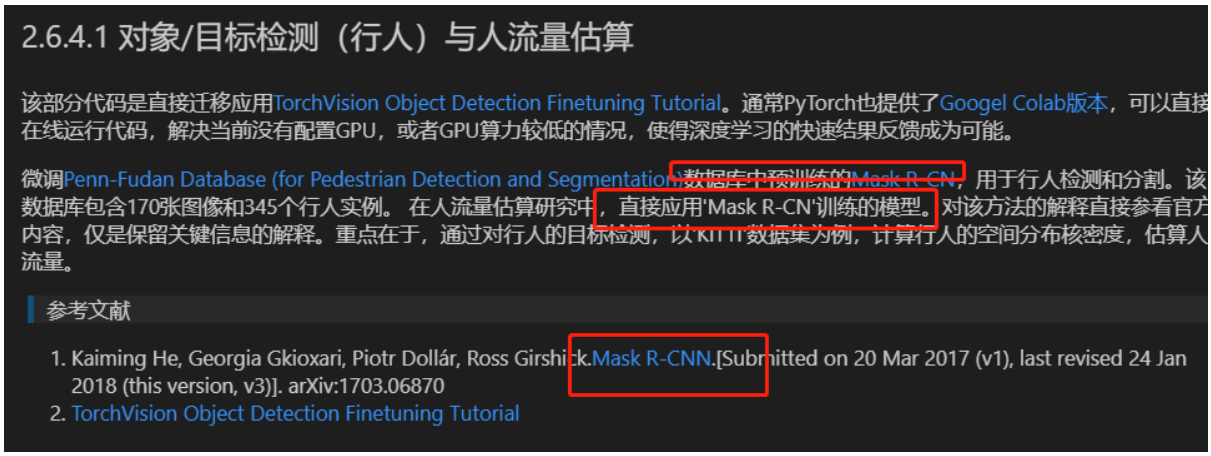
15. 标注的地方，函数方法字母缺少 r，【visualize_convFilter，visualize_convLayer】



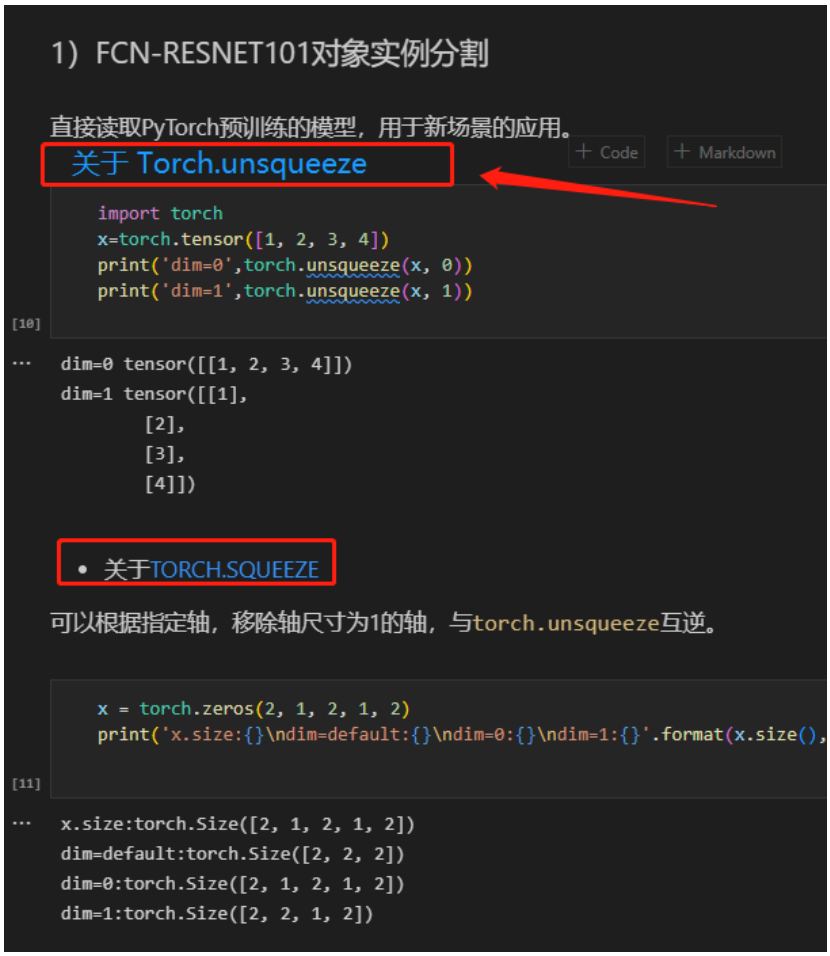
16. 2.6.4 导言，**对抗生产**改为**对抗生成**



17. 第一处和第二处，应该是 **mask R-CNN**



18. 可以也**添加一个小标题**，“**关于 TORCH.UNSQUEEZE**”，且最好先说 squeeze，再说 unsqueeze

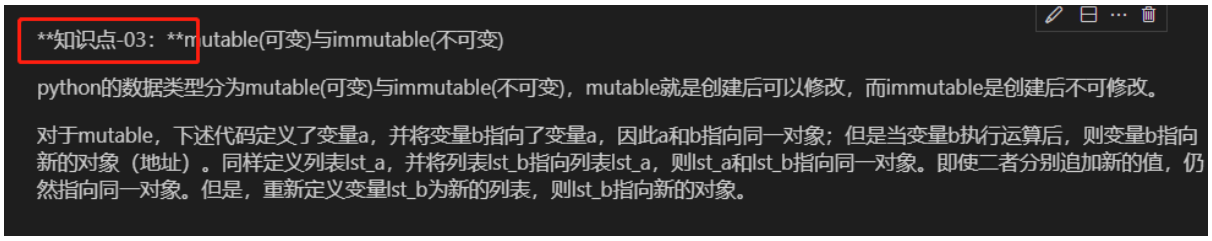


19. **2.6.5 的章节结构可以修改一下**

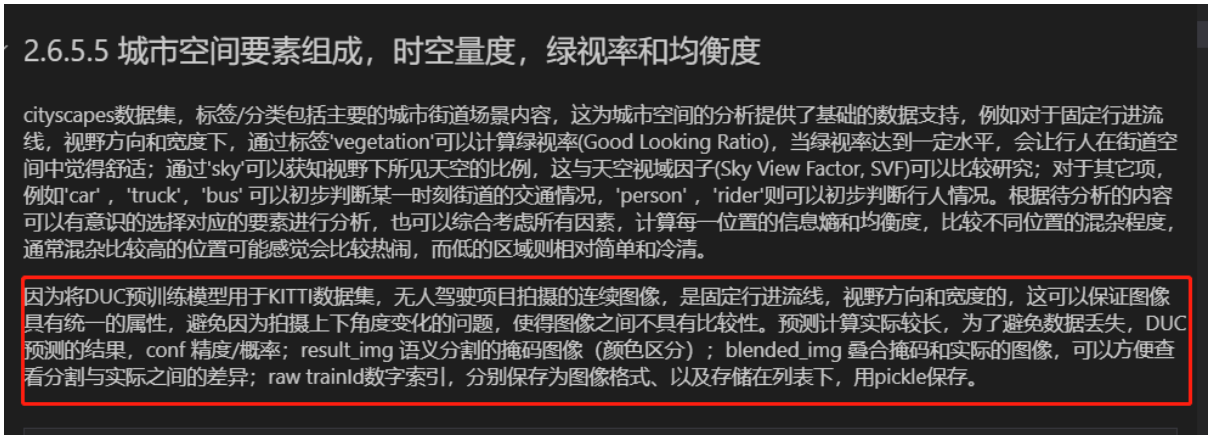
    可以在 2.6.5.1 之前，简要概述下，本章节的主要内容，即，介绍 Cityspcapes 数据集，并用其作为图像分割的数据集，然后对该数据集中的控件对象进行统计。最后提醒可能需要使用的知识点，**【2.6.6.1 之前也可以这样加一下】**

    其实可以将 2.6.5.1 知识点放在本章节的最后一部分，供读者查用学习。

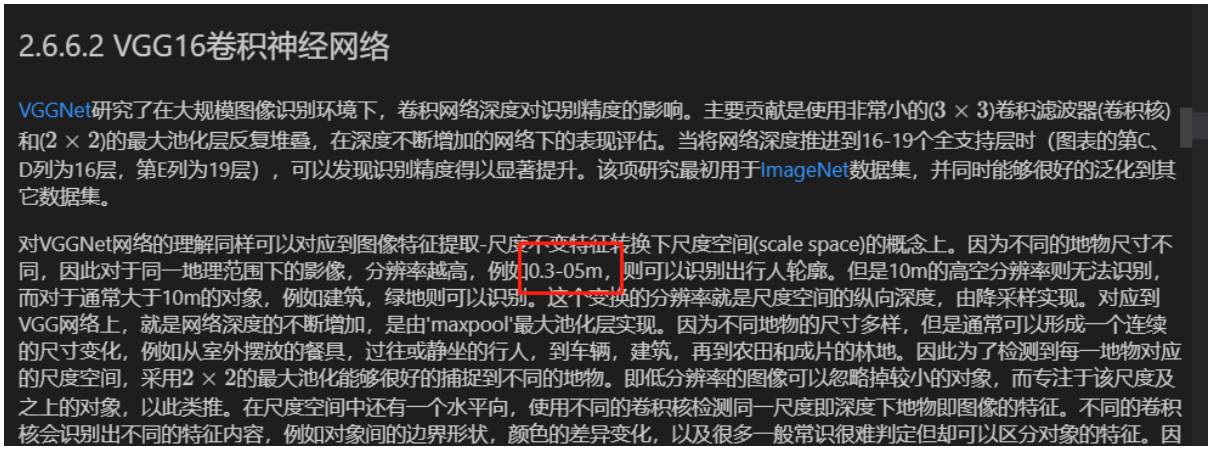
20. **知识点 03，显示格式问题，**可以在星号后添加空格



21. 2.6.5.5 的第二段，有点难以理解，可以重新理顺一下



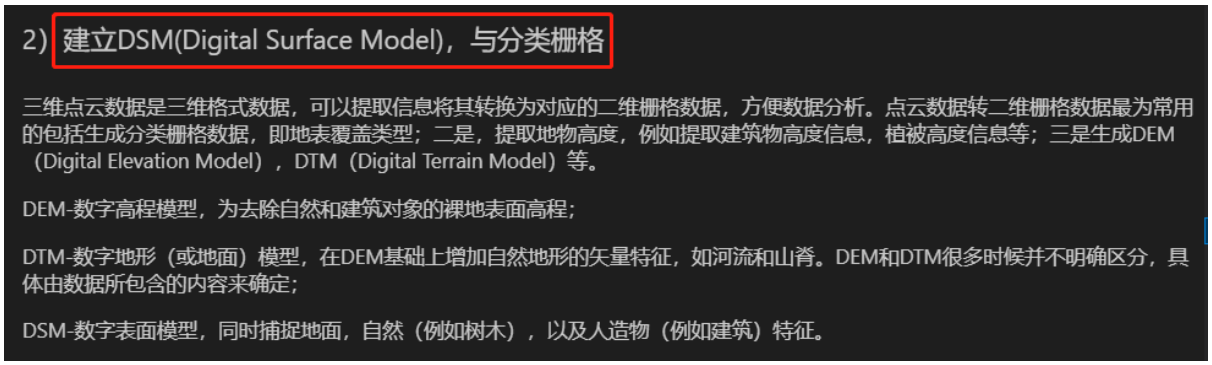
22. 0.3-0.5m，缺少小数点



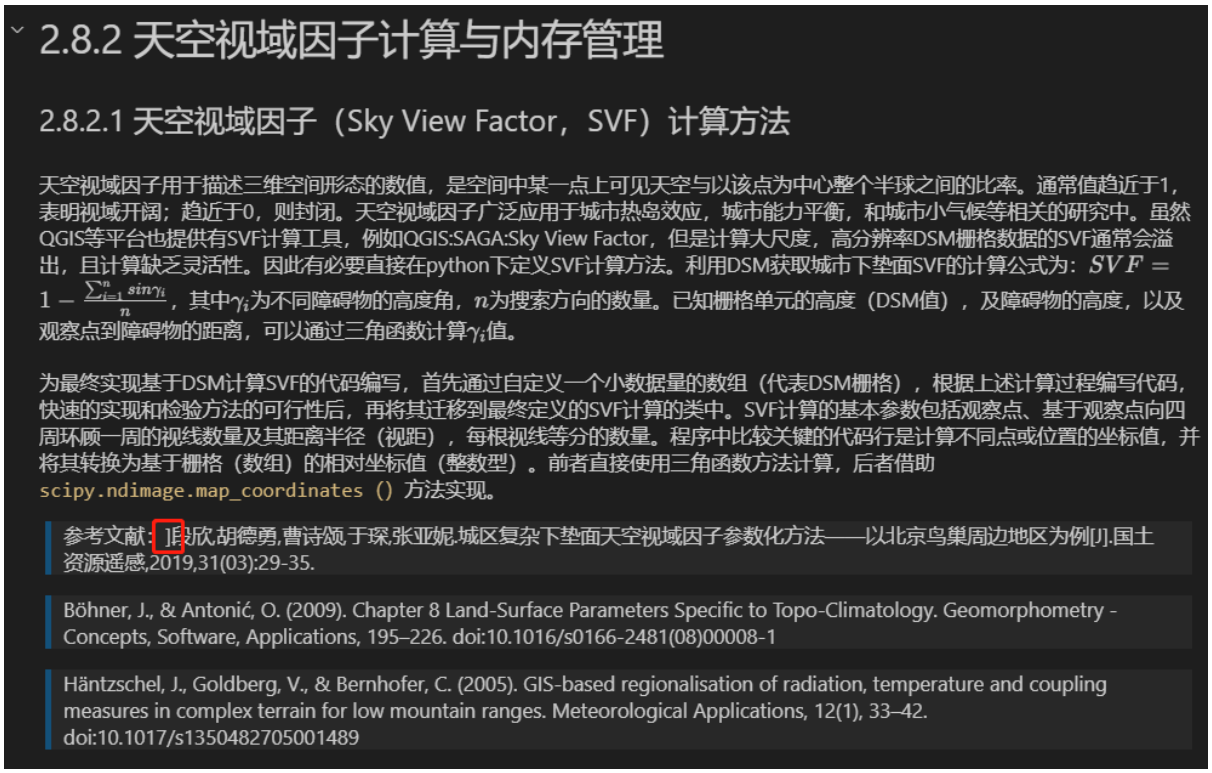
23. **小节标题可以修改调整一下**

建议改为，**建立数字表面模型（Digital Surface Model，DSM）与分类栅格**

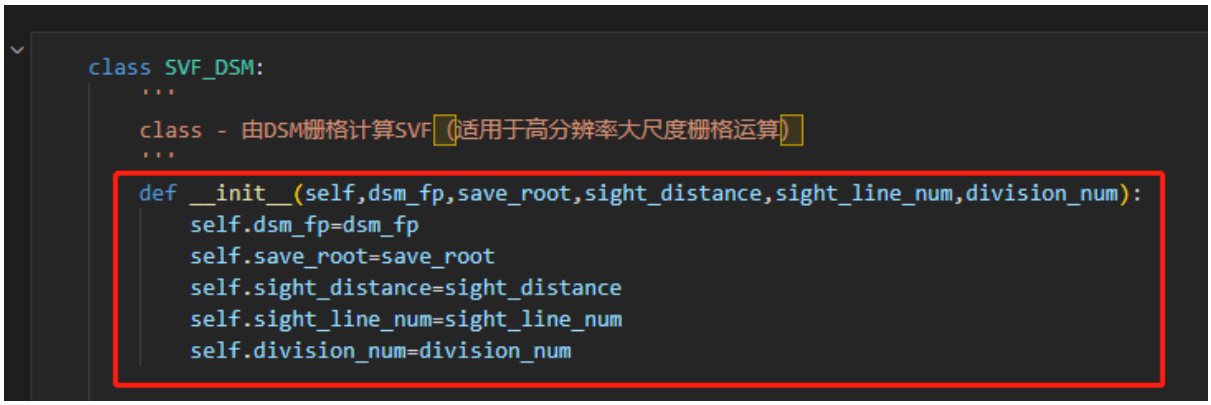
对于特殊名词，第一次出现应该先中文名称，然后括号里是英文全名，最后逗号隔开缩写，而对于一些英文名词缩写的介绍，可以先说全英文完整格式，然后翻译中文含义。



24. 2.8.2.1 中参考文献中格式问题



25. 定义类的时候，初始化默认参数时，最好也介绍下构造函数的参数含义



26. 代码中的 **print 帮助函数和注释**，还是存在**中英混用**的问题，我觉得为了有一致的统一性，且面向国内基础科研人员和本科生的话，中文或许会更合适一些

```
print()
print('-**-**-**-**-**-**- table data reading & selection  -**-**-**-**-**-')
# Read actual data from table. We are interested in collecting pressure values on entries where TDCcount fi
xs = [x for x in table.iterrows() if x['TDCcount'] > 3 and 20 <= x['pressure'] < 50]
pressure = [x['pressure'] for x in xs ]
print("Last record read:")
print(repr(xs[-1]))
print("Field pressure elements satisfying the cuts:")
print(repr(pressure))

# Read also the names with the same cuts
names = [x['name'] for x in table.where("""(TDCcount > 3) & (20 <= pressure) & (pressure < 50)""")]
print("Field names elements satisfying the cuts:")
print(repr(names))

print()
print('-**-**-**-**-**-**- array object creation  -**-**-**-**-**-**-**-')
print("Creating a new group called '/columns' to hold new arrays")
gcolumns = h5file.create_group(h5file.root, "columns", "Pressure and Name")

print("Creating an array called 'pressure' under '/columns' group")
h5file.create_array(gcolumns, 'pressure', np.array(pressure),"Pressure column selection")
print(repr(h5file.root.columns.pressure))

print("Creating another array called 'name' under '/columns' group")
h5file.create_array(gcolumns, 'name', names, "Name column selection")
print(repr(h5file.root.columns.name))

print("HDF5 file:")
print(h5file)

# Close the file
h5file.close()
print("File '" + filename + "' created")
```