

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

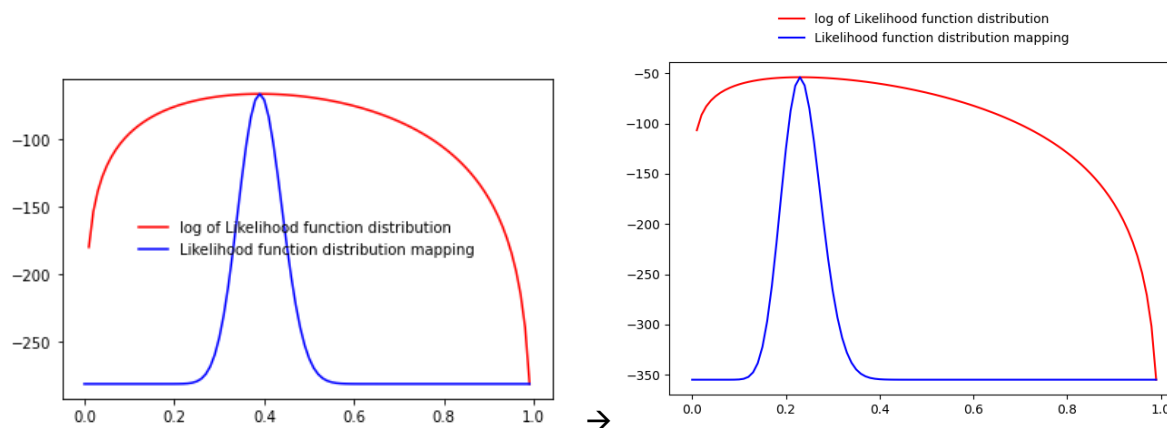
飞桨 **PaddlePaddle**-源于产业实践的开源深度学习平台

<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

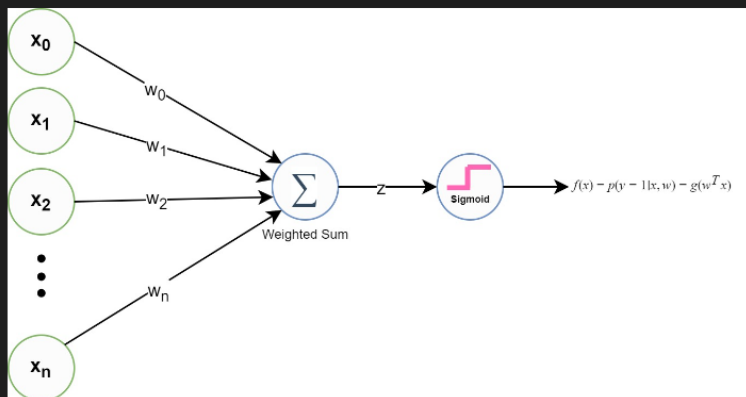


4. 2.6.2.4 中 alpha 学习率调整公式中，正文中未对 i 和 j 做出解释【见红色方框标注】

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： $p(x) = \sigma(t) = \frac{1}{1+e^{-t}} = \frac{1}{1+e^{-(\beta_0+\beta_1x_1+\beta_2x_2+\dots+\beta_nx_n)}} = \frac{1}{1+e^{-w^Tx}}$ ，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 $f(x) = \beta_0 + \beta_1x_1 + \beta_2x_2 + \dots + \beta_nx_n = w^Tx$ ，归一化到 [0,1] 之间。而因为二元分类，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 $\alpha = 4/(1.0+j+i) \times 0.0001$ 。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 $\text{randIndex} = \text{int}(\text{np.random.uniform}(0, \text{len}(\text{dataIndex})))$ 。

```
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： $\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$ for $j = 1, \dots, K$ 。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： $P(y=j|x) = \frac{e^{w_j^Tx}}{\sum_{k=1}^K e^{w_k^Tx}}$ ，这可以被视为 K 个线性函数 $x \mapsto x^T w_1, \dots, x \mapsto x^T w_K$ ，SoftMax 函数的复合。亦可以表述为，对于输入数据 $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ ，有 K 个类别，即 $y_i \in \{1, 2, \dots, k\}$ ，那么 SoftMax 回归主要估算输入数据 x_i 属于每一类的概率，即 $h_{\theta}(x_i) =$

$$\begin{bmatrix} p(y_1=1|x_i;\theta) \\ p(y_1=2|x_i;\theta) \\ \vdots \\ p(y_1=k|x_i;\theta) \end{bmatrix} = \frac{1}{\sum_{k=1}^K e^{\theta_k^T x_i}} \begin{bmatrix} e^{\theta_1^T x_i} \\ e^{\theta_2^T x_i} \\ \vdots \\ e^{\theta_k^T x_i} \end{bmatrix}, \text{ 其中 } \theta_1, \theta_2, \dots, \theta_k \in \theta \text{ 是模型的参数，乘以 } \frac{1}{\sum_{k=1}^K e^{\theta_k^T x_i}} \text{ 是为了让概}$$

率位于 [0,1] 并且概率之和为 1，SoftMax 回归将输入数据 x_i 属于类别 j 的概率为： $p(y_j=j|x_i;\theta) = \frac{e^{\theta_j^T x_i}}{\sum_{k=1}^K e^{\theta_k^T x_i}}$ 。

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，模型的取值分别被表述为 w 和 θ ，亦或者以矩阵的方式表述的方程组等。不管公式的表述方式如何，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 $1\{\cdot\}$ 函数，这种表达方式不是很通用常见

• 定义损失函数/交叉熵(cross entropy)损失函数——`loss(self,y_one_hot,y_probs)`

SoftMax运算符将输出变换为一个合法的类标预测分布（联合预测概率，预测概率分布），真实标签通过独热编码转换后的形式也是一种类别分布表达（0值为概率为0，1值为概率为1），二者形状保持一致。评估预测概率和真实标签的分布，只需要估计对应正确类别的预测概率。这样衡量两个概率分布差异的测量函数，可以使用交叉熵(cross entropy)，公式为： $H(y^{(i)}, \hat{y}^{(i)}) = -\sum_{j=1}^q y_j^{(i)} \log \hat{y}_j^{(i)}$ ，其中带下标的 $y_j^{(i)}$ 是向量 $y^{(i)}$ 中非0即1的元素（独热编码）； $\hat{y}_j^{(i)}$ 为联合预测概率 $\hat{y}^{(i)}$ 中对应类标（对应正确类别）的预测概率。

根据交叉熵的定义，SoftMax回归的损失函数（代价函数）计算公式为： $L(\theta) = -\frac{1}{m} [\sum_{i=1}^m \sum_{j=1}^k 1\{y_i = j\} \log \frac{e^{\theta_j^T x_i}}{\sum_{l=1}^k e^{\theta_l^T x_i}}]$ ，其中 m 为样本数， k 代表类标数， $1\{\bullet\}$ 是示性函数， $1\{\text{值为真的表达式}\}=1$ ， $1\{\text{值为假的表达式}\}=0$ 。SoftMax回归的输出值的形状为 (m,k) ，每一样本（总共 m 个样本）对应有 k （类标数）个概率值，每一类标对应的概率值为 p_k （联合预测概率）。 p_k 概率值位于 $[0,1]$ 之间，当取对数后，对应值转换到 $[-\inf, 0]$ 之间，概率越大的值，即越趋近于1，当取对数后其值越趋近于0，示性函数对应数据

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"

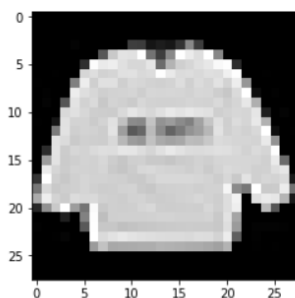
```
def fashionMNIST_show(imgs, labels, figsize=(12, 12)):
    import matplotlib.pyplot as plt
    from IPython import display
    import matplotlib_inline.backend_inline
    ...
    function - 打印显示Fashion-MNIST数据集图像
    ...

    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

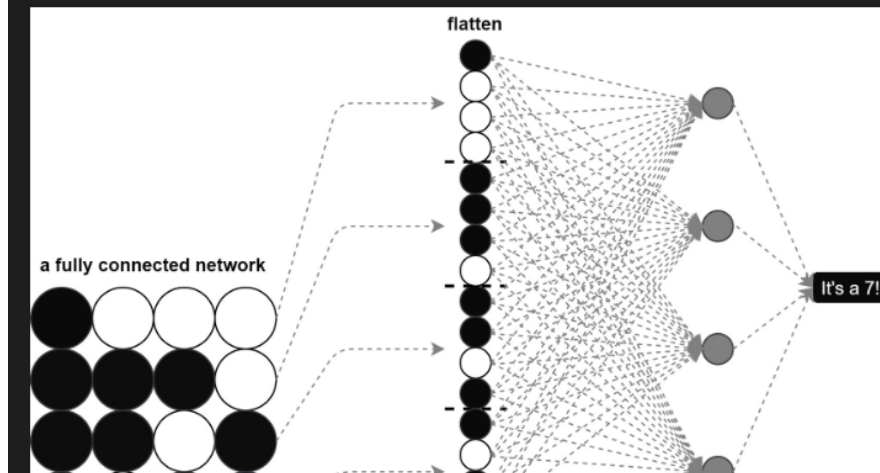
```
#将每批次样本x的形状转换为(batch_size,-1)
class flattenLayer(nn.Module):
    def __init__(self):
        super(flattenLayer,self).__init__()
    def forward(self,x):
        return x.view(x.shape[0],-1)
```

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, 来减少计算量, 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**)

卷积层和池化层都有一个**dilation**参数，可以翻译为**膨胀**。通过dilation配置，可以调整卷积核与图像的作用域，即感受野 (receptive field)。当 3×3 的卷积核dilation=1时，没有膨胀效应；当dilation=2时，感受野扩展至 7×7 ；当dilation=4时，感受野扩展至 15×15 。可以确定当dilation线性增加时，感受野时呈指数增加。

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

在设计卷积层，或者包含很多卷积层时，上述的方法可以进一步改进，定义一个函数可以一次性计算所有的卷积层。在输入参数设置时，使用了列表和元组的形式，固定输入参数'input'，'conv'和'pool'，值的参数依次为**[h_w, kernel_size, stride, pad, dilation]**。

```
conv_params=[
    ('input', (28, 28)),
    ('conv', [5, 1, 0, 1]), #h_w, kernel_size, stride, pad, dilation
    ('pool', [2, 2, 0, 1]),
    ('conv', [5, 1, 0, 1]),
    ('pool', [2, 2, 0, 1]),
]
```

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

2) 构建简单的卷积神经网络识别fashionMNIST数据集，与tensorboard

此处构建上图给出的神经网络结构，input-->conv1(relu)-->pool-->conv2(relu)-->pool-->fc1(relu)-->fc2(relu)-->fc3-->output，同时应用tensorboard可以写入并自动的根据写入的内容图示卷积神经网络结构，损失曲线，预测结果，样本信息，以及自定义的内容等。详细内容可以查看[torch.utils.tensorboard](#)。

参考: [Visualizing Models, Data, and Training with TensorBoard](#)

15. 标注的地方，函数方法字母缺少 r，**[visualize_convFilter, visualize_convLayer]**

3) 可视化卷积层/卷积核

通常一个卷积层的输出通道 (output channel) 有6, 16, 32, 64, 256, 等不确定的输出个数及更多的输出个数，即一个卷积层包含输出通道数目的filter/kernel过滤器/卷积核；而卷积核通过 3×3 , 5×5 , 7×7 , 等不确定尺寸 (通常为奇数) 或者更大尺寸来提取图像的特征，不同的卷积核提取的图像特征不同，或者表述为不同的卷积核关注不同的特征提取，这类似于图像的关键点描述子 (位于不同的尺度空间下)。不过因为卷积核的多样性，卷积提取的特征更加丰富多样。通过一个卷积层的多个卷积核 (尺度空间的水平向，表述各个像素值与各周边像素值的差异程度)，及多个卷积层 (尺度空间的纵深向，表述图像特征所在的 (对应的) 空间分辨率，例如遥感影像看清建筑轮廓的空间分辨率约为5-15m，看清行人的空间分辨率约为0.3-1m，而若要看清人的五官，空间分辨率则约为0.01-0.05m，这与对象 (特征) 的尺寸有关) 提取了大量的图像特征，将这些图像特征flatten展平，就构成了该图像的特征集合(feature maps)。

下述定义的conv_retriever类用于取回卷积神经网络中所有卷积层及其权重值 (卷积核)，取回的卷积层可以直接输入图像数据计算该卷积。例如上述网络取回的卷积层有两个。函数**visualize_convfilter**可以打印显示卷积核。函数**visualize_convLayer**则能打印显示指定数目的所有卷积图像结果。

参考 [Visualizing Filters and Feature Maps in Convolutional Neural Networks using PyTorch](#)

16. 2.6.4 导言，对抗生产改为对抗生成

2.6.4 对象检测、实例分割与人流量估算和对象统计

在影像和视频处理方面，深度学习主要涉及到图像分类 (Image classification)、对象检测 (Object detection)、迁移学习 (Transfer learning)、**对抗生成** (Adversarial generation: DCGAN/deep convolutional generative adversarial network) 等。在城市空间分析研究中，每一方向都能为之提供新的分析技术方法。对象检测是计算机视觉下，分析图像或影像，将其中的对象 (例如车辆、行人、动物、桌椅、植被等，通常包括室外环境、室内环境，甚至某一对象的具体再分，例如人脸识别等) 标记出来。通过图像对城市环境中对象的识别，可以统计对象的空间分布情况。同时，因为无人驾驶项目大量影像数据的获取，不仅可以对城市不连续的影像分析，亦可以通过连续影像的分析进一步获取目标对象的空间分布变化情况。在下文的分析中，对于对象的分析锁定在两个方向，一个是仅识别行人，实现人流量估计；再者是尽量多的识别对象，分析城市空间下各个对象的分布变化情况，以及对象之间的联系。

目前PyTorch已经集成了大量分析算法模型，以及提供预训练模型的参数，一定程度上可以跨过训练阶段，直接用于预测。PyTorch也提供了大量参考的代码 (还有大量的著作教程)，这都为研究者更快速和轻松的应用已有模型研究提供了便利，从而能够快速将其应用到各自的研究领域当中，而不是计算机视觉、深度学习算法本身研究的专业。

17. 第一处和第二处，应该是 **mask R-CNN**

2.6.4.1 对象/目标检测（行人）与人流量估算

该部分代码是直接迁移应用[TorchVision Object Detection Finetuning Tutorial](#)。通常PyTorch也提供了[Google Colab版本](#)，可以直接在线运行代码，解决当前没有配置GPU，或者GPU算力较低的情况，使得深度学习的快速结果反馈成为可能。

微调Penn-Fudan Database (for Pedestrian Detection and Segmentation)数据库中预训练的Mask R-CNN，用于行人检测和分割。该数据库包含170张图像和345个行人实例。在人流估算研究中，直接应用Mask R-CNN训练的模型。对该方法的解释直接参看官方内容，仅是保留关键信息的解释。重点在于，通过对行人的目标检测，以KITTI数据集为例，计算行人的空间分布核密度，估算人流量。

参考文献

1. Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick. **Mask R-CNN**. [Submitted on 20 Mar 2017 (v1), last revised 24 Jan 2018 (this version, v3)]. arXiv:1703.06870
2. [TorchVision Object Detection Finetuning Tutorial](#)

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

1) FCN-RESNET101对象实例分割

直接读取PyTorch预训练的模型，用于新场景的应用。

关于 Torch.unsqueeze

```
import torch
x=torch.tensor([1, 2, 3, 4])
print('dim=0',torch.unsqueeze(x, 0))
print('dim=1',torch.unsqueeze(x, 1))
```

[10]

```
... dim=0 tensor([[1, 2, 3, 4]])
    dim=1 tensor([[1],
                  [2],
                  [3],
                  [4]])
```

• 关于TORCH.SQUEEZE

可以根据指定轴，移除轴尺寸为1的轴，与torch.unsqueeze互逆。

```
x = torch.zeros(2, 1, 2, 1, 2)
print('x.size:{}\ndim=default:{}\ndim=0:{}\ndim=1:{}'.format(x.size(),
```

[11]

```
... x.size:torch.Size([2, 1, 2, 1, 2])
    dim=default:torch.Size([2, 2, 2])
    dim=0:torch.Size([2, 1, 2, 1, 2])
    dim=1:torch.Size([2, 2, 1, 2])
```

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 Cityscapes 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

知识点-03: mutable(可变)与immutable(不可变)

python的数据类型分为mutable(可变)与immutable(不可变), mutable就是创建后可以修改, 而immutable是创建后不可修改。

对于mutable，下述代码定义了变量a，并将变量b指向了变量a，因此a和b指向同一对象；但是当变量b执行运算后，则变量b指向新的对象（地址）。同样定义列表lst_a，并将列表lst_b指向列表lst_a，则lst_a和lst_b指向同一对象。即使二者分别追加新的值，仍然指向同一对象。但是，重新定义变量lst_b为新的列表，则lst_b指向新的对象。

21. 2.6.5.5 的第二段，有点难以理解，可以重新理顺一下

2.6.5.5 城市空间要素组成, 时空量度, 绿视率和均衡度

cityscapes数据集, 标签/分类包括主要的城市街道场景内容, 这为城市空间的分析提供了基础的数据支持, 例如对于固定行进流线, 视野方向和宽度下, 通过标签'vegetation'可以计算绿视率(Good Looking Ratio), 当绿视率达到一定水平, 会让行人在街道空间中觉得舒适; 通过'sky'可以获知视野下所见天空的比例, 这与天空视域因子(Sky View Factor, SVF)可以比较研究; 对于其它项, 例如'car', 'truck', 'bus'可以初步判断某一时刻街道的交通情况, 'person', 'rider'则可以初步判断行人情况。根据待分析的内容可以有意愿的选择对分析的要素进行析, 也可以综合考虑所有因素, 计算每一位置的信息熵和均衡度, 比较不同位置的混杂程度, 通常混杂比较高的位置可能感觉会比较热闹, 而低的区域则相对简单和冷清。

因为将DUC预训练模型用于KITTI数据集，无人驾驶项目拍摄的连续图像，是固定行进流线，视野方向和宽度的，这可以保证图像具有统一的属性，避免因为拍摄上下角度变化的问题，使得图像之间不具有比较性。预测计算实际较长，为了避免数据丢失，DUC预测的结果，conf 精度/概率；result_img 语义分割的掩码图像（颜色区分）；blended_img 叠合掩码和实际的图像，可以方便查看分割与实际之间的差异；raw trainId数字索引，分别保存为图像格式、以及存储在列表下，用pickle保存。

22. 0.3-0.5m, 缺少小数点

2.6.6.2 VGG16卷积神经网络

VGGNet研究了在大规模图像识别环境下，卷积网络深度对识别精度的影响。主要贡献是使用非常小的(3 × 3)卷积滤波器(卷积核)和(2 × 2)的最大池化层反复堆叠，在深度不断增加的网络下的表现评估。当将网络深度推进到16-19个全支持层时(图表的第C、D列为16层，第E列为19层)，可以发现识别精度得以显著提升。该项研究最初用于ImageNet数据集，并同时能够很好的泛化到其他数据集。

对VGGNet网络的理解同样可以对应到图像特征提取-尺度不变特征转换下尺度空间(scale space)的概念上。因为不同的地物尺寸不同, 因此对于同一地理范围下的影像, 分辨率越高, 例如0.3-0.5m, 则可以识别出行人轮廓。但是10m的高空分辨率则无法识别, 而对于通常大于10m的对象, 例如建筑, 绿地则可以识别。这个变换的分辨率就是尺度空间的纵向深度, 由降采样实现。对应到VGG网络上, 就是网络深度的不断增加, 是由'maxpool'最大池化层实现。因为不同地物的尺寸多样, 但是通常可以形成一个连续的尺寸变化, 例如从室外摆放的餐具, 过往或静坐的行人, 到车辆, 建筑, 再到农田和成片的林地。因此为了检测到每一地物对应的尺度空间, 采用 2×2 的最大池化能够很好的捕捉到不同的地物。即低分辨率的图像可以忽略掉较小的对象, 而专注于该尺度及之上的对象, 以此类推。在尺度空间中还有一个水平向, 使用不同的卷积核检测同一尺度即深度下地物即图像的特征。不同的卷积核会识别出不同的特征内容, 例如对象间的边界形状, 颜色的差异变化, 以及很多一般常识很难判定但却可以区分对象的特征。因

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。

2) 建立DSM(Digital Surface Model)，与分类栅格

三维点云数据是三维格式数据，可以提取信息将其转换为对应的二维栅格数据，方便数据分析。点云数据转二维栅格数据最为常用的包括生成分类栅格数据，即地表覆盖类型；二是，提取地物高度，例如提取建筑物高度信息，植被高度信息等；三是生成DEM (Digital Elevation Model)，DTM (Digital Terrain Model) 等。

DEM-数字高程模型，为去除自然和建筑对象的裸地表面高程；

DTM-数字地形（或地面）模型，在DEM基础上增加自然地形的矢量特征，如河流和山脊。DEM和DTM很多时候并不明确区分，具体由数据所包含的内容来确定；

DSM-数字表面模型，同时捕捉地面，自然（例如树木），以及人造物（例如建筑）特征。

24. 2.8.2.1 中参考文献中格式问题

2.8.2 天空视域因子计算与内存管理

2.8.2.1 天空视域因子（Sky View Factor, SVF）计算方法

天空视域因子用于描述三维空间形态的数值，是空间中某一点上可见天空与以该点为中心整个半球之间的比率。通常值趋近于1，表明视域开阔；趋近于0，则封闭。天空视域因子广泛应用于城市热岛效应，城市能力平衡，和城市小气候等相关的研究中。虽然QGIS等平台也提供有SVF计算工具，例如QGIS:SAGA:Sky View Factor，但是计算大尺度，高分辨率DSM栅格数据的SVF通常会溢出，且计算缺乏灵活性。因此有必要直接在python下定义SVF计算方法。利用DSM获取城市下垫面SVF的计算公式为： $SVF = 1 - \frac{\sum_{i=1}^n \sin \gamma_i}{n}$ ，其中 γ_i 为不同障碍物的高度角， n 为搜索方向的数量。已知栅格单元的高度（DSM值），及障碍物的高度，以及观察点到障碍物的距离，可以通过三角函数计算 γ_i 值。

为最终实现基于DSM计算SVF的代码编写，首先通过自定义一个小数据量的数组（代表DSM栅格），根据上述计算过程编写代码，快速的实现和检验方法的可行性后，再将其迁移到最终定义的SVF计算的类中。SVF计算的基本参数包括观察点、基于观察点向四周环顾一周的视线数量及其距离半径（视距），每根视线等分的数量。程序中比较关键的代码行是计算不同点或位置的坐标值，并将其转换为基于栅格（数组）的相对坐标值（整数型）。前者直接使用三角函数方法计算，后者借助 `scipy.ndimage.map_coordinates()` 方法实现。

参考文献：[1] 张欣, 胡德勇, 曹诗璇, 于琛, 张亚妮. 城区复杂下垫面天空视域因子参数化方法——以北京鸟巢周边地区为例[J]. 国土资源遥感, 2019, 31(03): 29-35.

Böhner, J., & AntoniĆ, O. (2009). Chapter 8 Land-Surface Parameters Specific to Topo-Climatology. *Geomorphometry - Concepts, Software, Applications*, 195–226. doi:10.1016/s0166-2481(08)00008-1

Häntzschel, J., Goldberg, V., & Bernhofer, C. (2005). GIS-based regionalisation of radiation, temperature and coupling measures in complex terrain for low mountain ranges. *Meteorological Applications*, 12(1), 33–42. doi:10.1017/s1350482705001489

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

```
class SVF_DSM:
    ...

    class - 由DSM栅格计算SVF (适用于高分辨率大尺度栅格运算)
    ...

    def __init__(self, dsm_fp, save_root, sight_distance, sight_line_num, division_num):
        self.dsm_fp = dsm_fp
        self.save_root = save_root
        self.sight_distance = sight_distance
        self.sight_line_num = sight_line_num
        self.division_num = division_num
```

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")
```